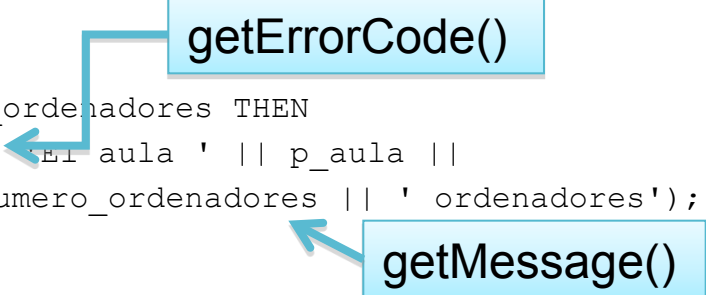


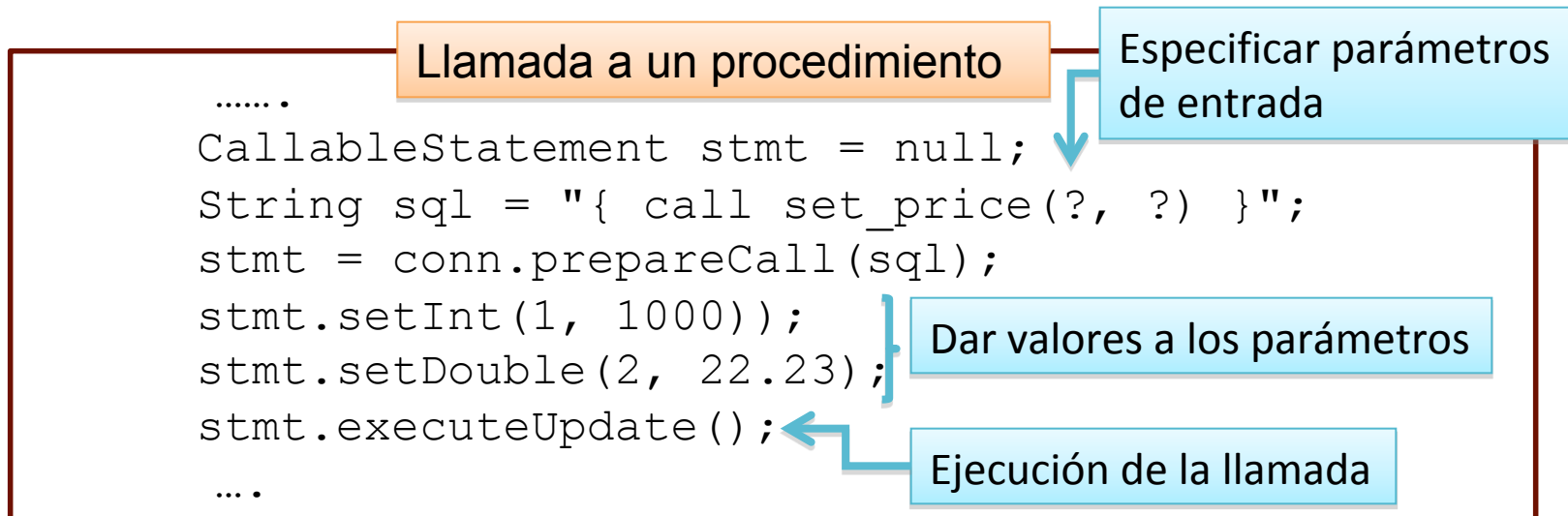
# Procedimientos almacenados y funciones

- En el desarrollo de aplicaciones es muy común que la lógica de negocio se implemente en el propio SGBD (la forma habitual de hacerlo en Oracle es mediante procedimientos, funciones y disparadores escritos en PL/SQL)
- Estos procedimientos se pueden utilizar, entre otras cosas, para realizar cálculos y operaciones
- Es una buena estrategia lanzar excepciones desde los procedimientos almacenados cuando se encuentra un error de lógica de negocio
- Dicha excepción debe enviar el mensaje de error para que sea capturado en el programa Java y tratada en la clase que hace la llamada al procedimiento almacenado

```
BEGIN
    ...
    IF v_numero_ordenadores > v_maximo_ordenadores THEN
        RAISE_APPLICATION_ERROR (-20001, 'El aula ' || p_aula ||
                                     'tiene ' || v_numero_ordenadores || ' ordenadores');
    END IF;
    ...
END;
```



- Para hacer una llamada a un procedimiento almacenado se utiliza un objeto de la clase **CallableStatement** que recibe el resultado de la invocación del método **prepareCall()** de la clase **Connection**. Previamente se construye la llamada al procedimiento o la función utilizando el símbolo '?' para indicar el lugar de los parámetros
- Para asignar valores a los parámetros de entrada se utiliza el método **setXXX()**, donde 'XXX' indica el tipo de datos del parámetro de entrada
- Finalmente se llama al método **executeUpdate()** para ejecutar la llamada al procedimiento o función

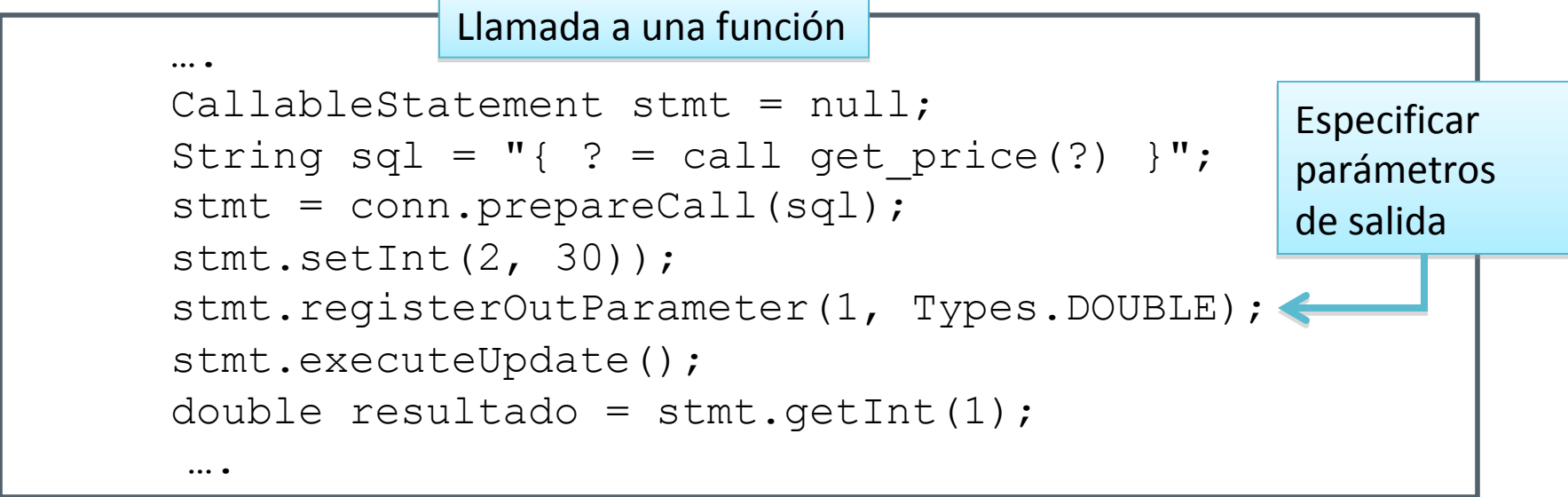


- Un procedimiento almacenado puede tener parámetros de entrada y **parámetros de salida**, que se utilizan para devolver información al programa. Especialmente, las funciones siempre tienen un parámetro de salida que es el valor que devuelve dicha función
- Para que un procedimiento almacenado o una función devuelva información mediante los parámetros de salida es necesario indicar, previamente, de qué tipo es cada parámetro mediante el método **registerOutParameter()**
- Para recuperar los valores de los parámetros de salida se utiliza el método **getXXX()**, donde '**XXX**' indica el tipo de datos del parámetro de salida

#### Llamada a una función

```
...  
CallableStatement stmt = null;  
String sql = "{ ? = call get_price(?) }";  
stmt = conn.prepareCall(sql);  
stmt.setInt(2, 30);  
stmt.registerOutParameter(1, Types.DOUBLE);  
stmt.executeUpdate();  
double resultado = stmt.getInt(1);  
...  
...
```

Especificar  
parámetros  
de salida



# Manejo de cursores

## Código PL/SQL del procedimiento almacenado

```
CREATE OR REPLACE PROCEDURE getDBUSERCursor(  
    p_username IN DBUSER.USERNAME%TYPE,  
    c_dbuser OUT SYS_REFCURSOR)  
IS  
BEGIN  
  
    OPEN c_dbuser FOR  
    SELECT * FROM DBUSER WHERE USERNAME LIKE p_username || '%';  
  
END;
```

## Fragmento de código Java para realizar una llamada al procedimiento almacenado

```
private static void callOracleStoredProcCURSORParameter()  
    throws SQLException {  
    String getDBUSERCursorSql = "{call getDBUSERCursor(?,?)}";  
    try {  
        CallableStatement call = conn.prepareCall(getDBUSERCursorSql);  
        call.setString(1, "Juan");  
        call.registerOutParameter(2, OracleTypes.CURSOR);  
        call.executeUpdate();  
        rs = (ResultSet) call.getObject(2);  
        while (rs.next()) {  
            String userid = rs.getString("USER_ID");  
            String userName = rs.getString("USERNAME");  
            String createdBy = rs.getString("CREATED_BY");  
            System.out.println("UserName : " + userid);  
            System.out.println("UserName : " + userName);  
            System.out.println("CreatedBy : " + createdBy);  
        }  
    } catch (SQLException e) {  
        System.out.println(e.getMessage());  
    } finally {  
        if (rs != null) rs.close();  
        if (call != null) call.close();  
    }  
}
```

Método **get** para capturar un cursor