



Excepciones - Base de datos

Excepciones



La clase Exception en Java es una clase base para las excepciones que se lanzan durante la ejecución de un programa.

Esta clase representa la mayoría de las excepciones que se producen en tiempo de ejecución, como las excepciones de índice fuera de rango, las excepciones de formato de datos, las excepciones de división por cero y muchas otras.

En resumen, la clase Exception en Java es la clase base para las excepciones que se producen durante la ejecución de un programa. Esta clase proporciona métodos y constructores para manejar excepciones y se extiende para crear subclases más específicas que se ajusten a situaciones particulares.

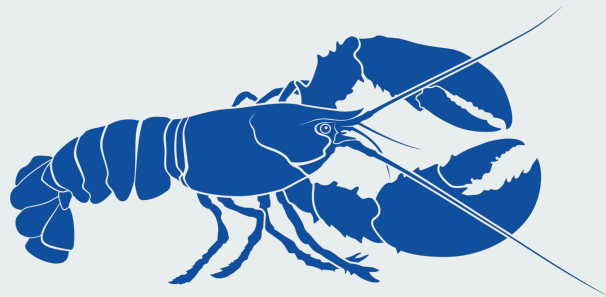
Excepciones

Para manejar excepciones,

En Java, el bloque try-catch es utilizado para manejar errores y excepciones que pueden ocurrir durante la ejecución del programa. El bloque try contiene el código que puede generar una excepción, mientras que el bloque catch se utiliza para manejar la excepción en caso de que se produzca.

La sintaxis básica es la siguiente:

```
}
```



Excepciones



```
try { // Código que puede generar una excepción

} catch (Excepcion1 e1) { // Manejar la excepción 1


} catch (Exception e) { // Manejar cualquier otra excepción

} finally { // Código que se ejecuta siempre, independientemente de si se produjo
una excepción o no

}

}
```

Excepciones



```
try{
    int resultado = 10 / 0; // Esto producirá una excepción de división por cero
}
catch (ArithmeticException e) {
    System.out.println("Error: " + e.getMessage()); // Manejar la excepción
    aritmética
}
finally {
    System.out.println("Esto se ejecutará siempre"); // Esto se imprimirá siempre,
    incluso si se produjo una excepción
}
```

Excepciones - Propias



Para crear tu propia excepción en Java, debes seguir los siguientes pasos:

1. Crear una clase que extienda la clase `Exception` o una subclase de `Exception`, como `RuntimeException`.
2. Agregar un constructor a la clase que reciba un mensaje de error o información adicional que se pueda usar para manejar la excepción.
3. Opcionalmente, agregar cualquier otro método que necesites para manejar la excepción.
4. Utilizar la palabra clave `throw` para lanzar la excepción en tu código cuando se produce la situación que justifica la excepción.
5. Manejar la excepción utilizando un bloque `try-catch` o declarando que el método en el que se produce la excepción `throws` la excepción.

Excepciones - Propias



Ejemplo:

```
public class MiExcepcion extends Exception {  
    public MiExcepcion(String mensaje) {  
        super(mensaje);  
    }  
    public void imprimirMensaje() {  
        System.out.println("Esta es mi excepción personalizada: " +  
        getMessage());  
    }  
    public void miMetodo() throws MiExcepcion {  
        throw new MiExcepcion("Se produjo un error en mi método");  
    }  
}
```

Excepciones - Propias



Ejemplo:

```
public static void main(String[] args) {  
    MiExcepcion miexcepcion = new MiExcepcion("Mi excepción");  
    try {  
        miexcepcion.miMetodo();  
    }  
    catch (MiExcepcion e) {  
        e.imprimirMensaje();  
    }  
}
```


Excepciones - Propias



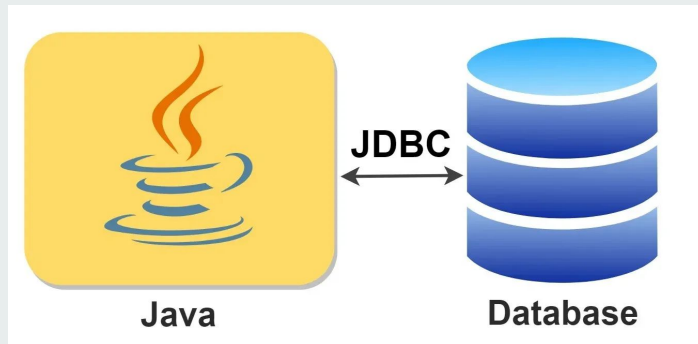
En resumen, la clase Exception en Java es la clase base para las excepciones que se producen durante la ejecución de un programa.

Esta clase proporciona métodos y constructores para manejar excepciones y se extiende para crear subclases más específicas que se ajusten a situaciones particulares.

JDBC

JDBC (Java Database Connectivity) es una API (Application Programming Interface) de Java que proporciona una interfaz estándar para interactuar con bases de datos relacionales.

bcJDBC permite a los desarrolladores de aplicaciones Java conectarse y manipular datos en una amplia variedad de bases de datos, como MySQL, Oracle, SQL Server, PostgreSQL, entre otros.



JDBC



La API JDBC consta de varias clases e interfaces que se utilizan para conectarse a una base de datos, enviar consultas y recibir resultados.

Algunas de las clases principales en JDBC incluyen

- DriverManager para manejar la conexión a la base de datos
- Connection para representar la conexión a la base de datos,
- Statement para enviar consultas a la base de datos
- ResultSet para representar los resultados de una consulta.

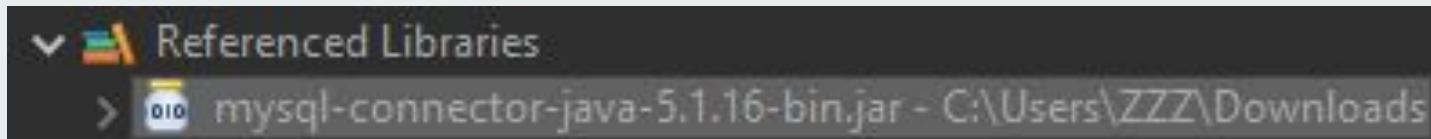
JDBC



Para utilizar JDBC en una aplicación Java, es necesario incluir el controlador JDBC correspondiente para la base de datos a la que se quiere conectar.

El controlador JDBC es un conjunto de clases y métodos específicos para la base de datos y se utiliza para proporcionar la conexión a la base de datos a través de JDBC.

Por ejemplo, para conectarse a una base de datos MySQL, se debe descargar el controlador JDBC de MySQL e incluirlo en el proyecto Java.



```
package Datos;  
import java.sql.Connection;  
import java.sql.DriverManager;
```

```
import javax.swing.JOptionPane;
```

```
public class Conexion {
```

```
    Connection con ;
```

```
    public Connection conectar() {
```

```
        try {
```

```
            Class.forName("com.mysql.jdbc.Driver");
```

```
            con =
```

```
            DriverManager.getConnection("jdbc:mysql://localhost:3306/alohanet","root","");
```

```
            JOptionPane.showMessageDialog(null, "se conecto");
```

```
        } catch (Exception e) {
```

```
            JOptionPane.showMessageDialog(null, "error al conectarse");
```

```
        }
```

```
        return con;
```

```
    }
```

Objeto connection

```
import java.sql.Connection;
```



Nos permite utilizar un objeto Connection

```
import java.sql.DriverManager;
```



Nos permite utilizar un controlador para la conexión.

```
import javax.swing.JOptionPane;
```



Librería que ya usamos para utilizar gráficamente mensajes

```
public class Conexion {
```

```
    Connection con ;
```



Creo mi objeto connection

```
}
```

Objeto connection



```
public Connection conectar() {  
    try {  
        Class.forName("com.mysql.jdbc.Driver");  
        con =  
DriverManager.getConnection("jdbc:mysql://localhost:3306//MiBase","root","");  
        JOptionPane.showMessageDialog(null, "se conecto");  
    } catch (Exception e) {  
        JOptionPane.showMessageDialog(null, "error al conectarse");  
    }  
    return con;  
}
```

Objeto connection



DriverManager.getConnection(

"jdbc:mysql://localhost:3306//",



especifica el protocolo JDBC y el número de puerto

"MiBase",



El nombre de mi base de datos

"root",



El usuario de esta bbdd

""");



Este parámetro es la contraseña de la base de datos en este caso está vacía

Fin de la presentación



Temas Vistos:

Excepciones

Crear Excepciones

JDBC

Conexión con bbdd