Junio de 2024

INFORME.

Taller de Aplicaciones JAVA

Luis Fernando Meneses Gelvez

27.111.500-8

INGENIERÍA EN INFORMÁTICA.

Asignatura:

Taller de aplicaciones JAVA.

Profesor:

Sabina Romero.

Clase Conexión.

- **Importación de Clases**: la declaración **import** se utiliza para incluir clases específicas de otros paquetes en el código, lo que te permite utilizar esas clases sin tener que referenciarlas con el nombre de paquete completo cada vez.

```
import java.sql.PreparedStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
```

- **PreparedStatement**: Objeto que representa una declaración SQL precompilada.

Una declaración SQL se precompila y almacena en un PreparedStatementobjeto. Luego, este objeto se puede utilizar para ejecutar esta declaración de manera eficiente varias veces, permitiendo la ejecución de sentencias SQL parametrizadas.

- **Connection**: Representa una conexión a una base de datos. A través de este objeto, se pueden ejecutar comandos SQL y gestionar transacciones.
- **DriverManager**: Clase que maneja un conjunto de controladores de base de datos.
- **SQLException**: Clase que maneja excepciones relacionadas con SQL.

Detalles de la conexión a la base de datos:

- **JDBC_DRIVER**: Especifica el nombre del driver JDBC necesario para conectar con MySQL.
- **DB_URL**: URL de la base de datos MySQL.
- **USER**: Nombre de usuario para la base de datos.
- **PASS**: Contraseña del usuario.

```
public static String JDBC_DRIVER = "com.mysql.jdbc.Driver";
public static String DB_URL = "jdbc:mysql://localhost/registro_atenciones";
public static String USER = "root";
public static String PASS = "ipchile";
```

Establecer la conexión:

Creamos el método **conectar()** para conectarnos a la base de datos. Este método devolverá un objeto de tipo Connection.

Este método se encarga de registrar el Diver JDBC, establecer una conexión con la base de datos utilizando las credenciales proporcionadas y mostrar un mensaje si la conexión se estableció con éxito.

```java
public static void conectar(){

try{
    // Registrar el driver JDBC
    Class.forName(className:JDBC_DRIVER);
    // Establecer la conexión con la base de datos usando las credenciales proporcionadas
    conx = DriverManager.getConnection(url: DB_URL, user:USER, password: PASS);
    if(conx!=null){
        System.out.println("Conexion Establecida"+DB_URL);
    }
} catch ( SQLException e) {
    // Manejar cualquier excepción
    System.out.println(x: "Hubo un Error al Conectar");
}catch(ClassNotFoundException ex){
    System.out.println(x: ex);
}
}
public static void desconectar() throws SQLException{
    conx.close();
}
```

Manejo de Excepciones:

El manejo de excepciones en Java es fundamental, utilizando bloques try, catch y finally.

El bloque try se utiliza para envolver el código que puede generar una excepción. Si se produce una excepción dentro del bloque try, se captura en uno o más bloques catch asociados.

Por otro lado, el bloque finally se utiliza para ejecutar código que debe ejecutarse, independientemente de si se ha producido una excepción o no. Se usa para liberar recursos que deben cerrarse, como conexiones a bases de datos o archivos.

Throws: La palabra reservada o cláusula throws permite indicar que el método puede lanzar una o más excepciones. Por tanto tiene que ir declarada en el método.

No lanza la excepción directamente, sino que declara que el método podría arrojarla.

Puede arrojar una excepción de tipo SQLException. Esto significa que si ocurre algún error al cerrar la conexión (conx.close() se lanzará una excepción de tipo SQLException.

Clase Pacientes.

Esta clase representa un modelo de paciente que contiene atributos, en Java, también conocidos como campos o variables de instancia, son variables que pertenecen a una clase y representan las características o propiedades que describen el estado de un objeto.

los atributos pueden clasificarse en varios tipos:

Atributos de instancia, Atributos estáticos, Atributos finales, Atributos de clase.

```
public class Pacientes {

    private String rut_p;
    private String nombre_p;
    private String apellidos_p;
    private String email;
```

Atributos De Instancia

Constructores: es un método especial que se utiliza para inicializar un objeto recién creado y asignarle valores iniciales a sus variables de instancia.

En otras palabras, un constructor es una función que se ejecuta automáticamente cuando se crea un objeto de una clase específica, son **métodos especiales** que se utilizan para **inicializar objetos** cuando se crean instancias de una clase.

Constructor con parámetros: asigna valores a los atributos correspondientes de la clase.

```
public Pacientes() {
}

public Pacientes(String rut_p, String nombre_p, String apellidos_p, String email) {
    this.rut_p = rut_p;
    this.nombre_p = nombre_p;
    this.apellidos_p = apellidos_p;
    this.email = email;
}
```

Métodos:

- Utilizan la clase Conexión para establecer y cerrar la conexión con la base de datos.

- Preparan y ejecutan consultas SQL para insertar, seleccionar, actualizar y eliminar datos de la tabla "Pacientes".
- Manejan las excepciones de SQL para gestionar posibles errores que ocurran durante las operaciones.
- **strSQL:** es una variable de tipo String que se usa para almacenar la sentencia que inserta datos en la base de datos.
- **Conexion.conectar():** se utiliza para llamar al método conectar() de la clase Conexion para establecer una conexión con la base de datos.
- **PreparedStatement:** se utiliza para ejecutar la consulta parametrizada.
- **Conexion.st.executeUpdate():** Ejecuta la sentencia SQL para insertar los datos en la base de datos.

```
public void guardarPaciente() throws SQLException {
    try {
        String strSQL = "INSERT INTO Pacientes VALUES (?, ?, ?, ?)";
        Conexion.conectar(); //llamo al metodo conectar
        Conexion.st = Conexion.conx.prepareStatement(string: strSQL);
        Conexion.st.setString(i: 1, string: rut_p);
        Conexion.st.setString(i: 2, string: nombre_p);
        Conexion.st.setString(i: 3, string: apellidos_p);
        Conexion.st.setString(i: 4, string: email);
        Conexion.st.executeUpdate();
        System.out.println(x: "Datos del paciente almacenados correctamente.");


    } catch (SQLException e) {
        System.out.println("Error al guardar el paciente: " + e.getMessage());
    } finally {
        Conexion.desconectar();
    }
}
```

Los métodos **Get** y **Set**, son simples métodos que usamos en las clases para mostrar (**get**) o modificar (**set**) el valor de un atributo.

```
public String getNombre_p() {
    return nombre_p;
}

public void setNombre_p(String nombre_p) {
    this.nombre_p = nombre_p;
}
```

Clase Interfaz.

El método main es el punto de partida de la aplicación y se encarga de iniciar la interfaz gráfica para que el usuario pueda interactuar con ella. Aquí se crea una instancia de la clase Interfaz que representa una ventana y la hace visible para el usuario.

```
public static void main(String args[]) {

    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Interfaz().setVisible(b: true);
        }
    });
}
```

Eventos en java:

Los eventos de usuario son acciones o sucesos que ocurren durante la interacción del usuario con una interfaz de usuario, como hacer clic en un botón, presionar o soltar una tecla.

Métodos:

Estos métodos se ejecutan cuando se hace clic en un botón de la ventana, llamando al método asociado de la clase Pacientes.

En este ejemplo se Instancia objPac de la clase pacientes, para almacenar los datos en los atributos de esa instancia, luego, invocamos el método guardarPaciente() en esa instancia, lo que permite guardar los datos del paciente en la base de datos.

Luego se crea una variable llamada pacientesInfo de tipo String, se llama al método estático mostrarTodos() de la clase Pacientes, este método consulta la base de datos y obtiene información sobre todos los pacientes registrados, el cual será mostrado a través de un jTextArea1.

```java
private void btGuardarPActionPerformed(java.awt.event.ActionEvent evt) {
                //instancio un nuevo objeto
Pacientes objPac = new Pacientes(rut_p: this.jText_rut.getText(), nombre_p: this.jText_nombre.getText(),

try {
    objPac.guardarPaciente(); // Guardar el paciente en la base de datos
    // Obtener la lista actualizada de todos los pacientes y mostrarla en el JTextArea
    String pacientesInfo = Pacientes.mostrarTodos();
    this.jTextArea1.setText("Datos Actualizados.\n\n" + pacientesInfo);

    jText_rut.setText(t: "");
    jText_nombre.setText(t: "");
    jText_apellido.setText(t: "");
    jText_correo.setText(t: "");

} catch (SQLException e) {
    // Manejar excepciones de SQL
    System.out.println("Error al guardar paciente en la base de datos: " + e.getMessage());

} catch (Exception e) {
    // Manejar cualquier otra excepción
    System.out.println("Error al guardar paciente: " + e.getMessage());
```

Conclusión.

La conexión a la base de datos en Java se realiza a través de la API JDBC, que proporciona un medio para interactuar con bases de datos como MySQL. Para establecer y gestionar esta conexión, es fundamental manejar adecuadamente variables, métodos y constructores.

La clase **Conexión** maneja de manera efectiva la interacción con la base de datos MySQL. Utiliza objetos clave como Connection, DriverManager, y PreparedStatement, que garantizan conexiones seguras y eficientes. El método conectar(), que registra el driver JDBC y establece la conexión utilizando credenciales específicas, es esencial para iniciar la comunicación con la base de datos.

El manejo de excepciones es mediante bloques try y catch asegura la gestión correcta de recursos, minimizando riesgos y garantizando la estabilidad de la aplicación.

Bibliografía.

https://openwebinars.net/blog/introduccion-a-poo-en-java-excepciones/##manejo-de-excepciones-en-java

https://www.webferrol.com/diferencias-entre-throw-y-throws/

https://openwebinars.net/blog/introduccion-a-poo-en-java-atributos-y-constructores/

https://blog.hubspot.es/website/que-es-constructor-java

https://www.discoduroderoer.es/metodos-get-y-set-en-java/#google_vignette