TRABAJO PRÁCTICO NRO 4
SEMINARIO DE PRÁCTICA
DE INFORMATICA

Sistema de Gestión Integral
para una Clínica de Salud

Antonella Diaz

DNI 28.910.424

VINFOR12606

Índice

Desarrollo:

Para continuar con el desarrollo del Sistema de Gestión Integral para Clínicas de Salud, vamos a incluir una selección de patrón de diseño, persistencia y consulta de datos en una base de datos MySQL, correcta aplicación de excepciones para la interacción con la base de datos, inclusión de clases abstractas o interfaces y utilización de arreglos y la clase ArrayList.

1.Selección del Patrón de Diseño y Justificación

Patrón de Diseño: DAO (Data Access Object)

Justificación: El patrón DAO es adecuado para separar la lógica de negocio de la lógica de acceso a datos. Permite que los objetos de negocio accedan a los datos sin conocer los detalles de la implementación del almacenamiento. Este patrón también facilita el mantenimiento y escalabilidad del sistema, ya que los cambios en la lógica de acceso a datos no afectarán a la lógica de negocio.

2.Persistencia y Consulta de Datos en una Base de Datos MySQL

Configuración del Entorno

Primero, necesitamos agregar la biblioteca de MySQL Connector a nuestro proyecto. Esto se puede hacer descargando el archivo JAR del conector MySQL y añadiéndolo al classpath del proyecto.

Dependencia Maven :

En XML

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.27</version>
</dependency>
```

Configuración de la Base de Datos MySQL

Creamos una base de datos MySQL llamada clinica y las tablas necesarias para almacenar la información de pacientes, médicos, citas, inventarios y facturación.

En SQL

```
CREATE DATABASE clinica;

USE clinica;

CREATE TABLE Paciente (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100),
    direccion VARCHAR(255),
```

```
    telefono VARCHAR(20),
    numeroHistoriaClinica VARCHAR(50)
);

CREATE TABLE Medico (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100),
    direccion VARCHAR(255),
    telefono VARCHAR(20),
    especialidad VARCHAR(100)
);

CREATE TABLE Cita (
    id INT AUTO_INCREMENT PRIMARY KEY,
    fecha DATE,
    pacienteId INT,
    medicoId INT,
    FOREIGN KEY (pacienteId) REFERENCES Paciente(id),
    FOREIGN KEY (medicoId) REFERENCES Medico(id)
);

CREATE TABLE Producto (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100),
    cantidad INT,
    proveedor VARCHAR(100),
    fechaVencimiento DATE
);

CREATE TABLE Factura (
    id INT AUTO_INCREMENT PRIMARY KEY,
    numeroFactura VARCHAR(50),
    pacienteId INT,
    monto DOUBLE,
    FOREIGN KEY (pacienteId) REFERENCES Paciente(id)
);
```

Código Java para Conexión y Manejo de Base de Datos

Clase de Conexión a la Base de Datos:

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

public class DatabaseConnection {
    private static final String URL = "jdbc:mysql://localhost:3306/clinica";
    private static final String USER = "root";
    private static final String PASSWORD = "password";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

DAO Interface y Clase Abstracta:

```
import java.util.List;

public interface DAO<T> {
    void insertar(T t) throws SQLException;
    T obtener(int id) throws SQLException;
    List<T> obtenerTodos() throws SQLException;
    void actualizar(T t) throws SQLException;
    void eliminar(int id) throws SQLException;
}
```

```
public abstract class AbstractDAO<T> implements DAO<T> {
    protected Connection connection;

    public AbstractDAO() {
        try {
            this.connection = DatabaseConnection.getConnection();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

DAO de Paciente:

```
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class PacienteDAO extends AbstractDAO<Paciente> {
    @Override
    public void insertar(Paciente paciente) throws SQLException {
        String query = "INSERT INTO Paciente (nombre, direccion, telefono,
numeroHistoriaClinica) VALUES (?, ?, ?, ?)";
        PreparedStatement ps = connection.prepareStatement(query);
        ps.setString(1, paciente.getNombre());
        ps.setString(2, paciente.getDireccion());
        ps.setString(3, paciente.getTelefono());
        ps.setString(4, paciente.getNumeroHistoriaClinica());
        ps.executeUpdate();
    }

    @Override
    public Paciente obtener(int id) throws SQLException {
        String query = "SELECT * FROM Paciente WHERE id = ?";
        PreparedStatement ps = connection.prepareStatement(query);
        ps.setInt(1, id);
        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            return new Paciente(
                rs.getString("nombre"),
                rs.getString("direccion"),
                rs.getString("telefono"),
                rs.getString("numeroHistoriaClinica")
            );
        }
        return null;
    }

    @Override
    public List<Paciente> obtenerTodos() throws SQLException {
        String query = "SELECT * FROM Paciente";
        PreparedStatement ps = connection.prepareStatement(query);
        ResultSet rs = ps.executeQuery();
        List<Paciente> pacientes = new ArrayList<>();
        while (rs.next()) {
            pacientes.add(new Paciente(
                rs.getString("nombre"),
                rs.getString("direccion"),
                rs.getString("telefono"),
                rs.getString("numeroHistoriaClinica")
            ));
        }
        return pacientes;
    }

    @Override
    public void actualizar(Paciente paciente) throws SQLException {
        String query = "UPDATE Paciente SET nombre = ?, direccion = ?, telefono = ?,
numeroHistoriaClinica = ? WHERE id = ?";
        PreparedStatement ps = connection.prepareStatement(query);
        ps.setString(1, paciente.getNombre());
        ps.setString(2, paciente.getDireccion());
        ps.setString(3, paciente.getTelefono());
        ps.setString(4, paciente.getNumeroHistoriaClinica());
```

```
        ps.setInt(5, paciente.getId());
        ps.executeUpdate();
    }

    @Override
    public void eliminar(int id) throws SQLException {
        String query = "DELETE FROM Paciente WHERE id = ?";
        PreparedStatement ps = connection.prepareStatement(query);
        ps.setInt(1, id);
        ps.executeUpdate();
    }
}
```

3.Correcta Aplicación de Excepciones

Al interactuar con la base de datos, es crucial manejar adecuadamente las excepciones para asegurar la robustez del sistema. Utilizaremos bloques try-catch para manejar SQLException y otras posibles excepciones.

Ejemplo de Manejo de Excepciones en PacienteDAO:

```
public class PacienteDAO extends AbstractDAO<Paciente> {
    @Override
    public void insertar(Paciente paciente) {
        String query = "INSERT INTO Paciente (nombre, direccion, telefono,
numeroHistoriaClinica) VALUES (?, ?, ?, ?)";
        try (PreparedStatement ps = connection.prepareStatement(query)) {
            ps.setString(1, paciente.getNombre());
            ps.setString(2, paciente.getDireccion());
            ps.setString(3, paciente.getTelefono());
            ps.setString(4, paciente.getNumeroHistoriaClinica());
            ps.executeUpdate();
        } catch (SQLException e) {
            System.err.println("Error al insertar paciente: " + e.getMessage());
        }
    }

    @Override
    public Paciente obtener(int id) {
        String query = "SELECT * FROM Paciente WHERE id = ?";
        try (PreparedStatement ps = connection.prepareStatement(query)) {
            ps.setInt(1, id);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                return new Paciente(
                    rs.getString("nombre"),
                    rs.getString("direccion"),
                    rs.getString("telefono"),
                    rs.getString("numeroHistoriaClinica")
                );
            }
        } catch (SQLException e) {
            System.err.println("Error al obtener paciente: " + e.getMessage());
        }
        return null;
    }

    // Implementación de otros métodos con manejo de excepciones similar...
}
```

4.Inclusión de Clases Abstractas o Interfaces

Ya hemos incluido interfaces y clases abstractas para definir la estructura del DAO y asegurar la implementación de métodos necesarios.

5.Utilización Complementaria de Arreglos y de la Clase ArrayList

Usaremos ArrayList para manejar listas de objetos dinámicamente, mientras que los arreglos pueden ser utilizados para operaciones específicas donde el tamaño es fijo.

Ejemplo de uso de ArrayList en Inventario:

```
public class Inventario {
    private ArrayList<Producto> productos;

    public Inventario() {
        productos = new ArrayList<>();
    }

    public void agregarProducto(Producto producto) {
        productos.add(producto);
    }

    public void eliminarProducto(Producto producto) {
        productos.remove(producto);
    }

    public Producto buscarProducto(String nombre) {
        for (Producto producto : productos) {
            if (producto.getNombre().equalsIgnoreCase(nombre)) {
                return producto;
            }
        }
        return null;
    }

    public void mostrarInventario() {
        for (Producto producto : productos) {
            System.out.println(producto);
        }
    }
}
```

Ejemplo de uso de Arreglos:

Uso de Arreglo para una Operación Específica:

```
public class EstadisticasInventario {
    public static double calcularPromedioCantidad(Producto[] productos) {
        int total = 0;
        for (Producto producto : productos) {
            total += producto.getCantidad();
        }
        return total / (double) productos.length;
    }
}
```

Conclusión:

Con la implementación del patrón DAO, manejo de excepciones, inclusión de clases abstractas e interfaces, y el uso complementario de arreglos y ArrayList, se puede desarrollar un Sistema de Gestión Integral para Clínicas de Salud robusto y escalable. Este sistema no solo facilitará la gestión de datos y operaciones de la clínica, sino que también mejorará la eficiencia y la calidad del servicio brindado a los pacientes.