**Tiempo peor y mejor esperado**

**Clase Cliente\_beans**

**Metodo incremento\_Cliente**

public int incremento\_Cliente() throws SQLException{

// Tiempo mejor esperado: O(1) (Cuando la tabla cliente está vacía)

// Tiempo peor esperado: O(n) (Cuando la tabla cliente tiene n registros)

int inc=0;

ResultSet rs;

rs = bd.consultaBD("SELECT max(id\_cliente) as num FROM cliente;");

if (rs.next()){

inc = rs.getInt(1)+1;

}else{

inc = 1;

}

return inc;

}

**Metodo insertar\_cliente**

public void insertar\_cliente() throws SQLException{

// Tiempo mejor esperado: O(1) (Suponiendo que la inserción es rápida)

// Tiempo peor esperado: O(1) (Suponiendo que la inserción es rápida)

try{

String cadena = "insert into cliente values ('" + incremento\_Cliente() + "','" + getNombres() + "','" + getApellidos() + "','" + getCedula() + "','" + getDireccion() + "','" + getCorreo() + "','" + getTelefono() + "')";

bd.actualizarBD(cadena);

} catch (SQLException e){

}

}

**Metodo eliminar\_cliente**

public void eliminar\_cliente(String idCliente) throws SQLException{

// Tiempo mejor esperado: O(1) (Suponiendo que la eliminación es rápida)

// Tiempo peor esperado: O(n) (Cuando la tabla cliente tiene n registros)

String sql = "DELETE FROM cliente WHERE id\_cliente = " + idCliente;

bd.actualizarBD(sql);

}

**Metodo consultaTabla**

public ResultSet consultaTabla(String sql) throws SQLException{

// Tiempo mejor esperado: O(1) (Suponiendo que la consulta es rápida)

// Tiempo peor esperado: Dependiendo de la complejidad de la consulta SQL

return bd.consultaBD(sql);

}

**Metodo consultar\_cliente**

public void consultar\_cliente() throws SQLException{

// Tiempo mejor esperado: O(1) (Cuando no hay registros en la tabla cliente)

// Tiempo peor esperado: O(n) (Cuando la tabla cliente tiene n registros)

ResultSet rs;

rs = bd.consultaBD("select \* from cliente");

while (rs.next()){

System.out.print(rs.getInt(1)+ " ");

System.out.print(rs.getString(2)+ " ");

System.out.print(rs.getString(3) + " ");

System.out.print(rs.getString(4) + " ");

System.out.print(rs.getString(5)+ " ");

System.out.print(rs.getString(6)+ " ");

System.out.print(rs.getString(7)+ " ");

System.out.println("");

}

}

**Clase Detalle\_beans**

**Metodo incremento\_detalle**

public int incremento\_detalle() throws SQLException {

// Tiempo mejor esperado: O(1) (Cuando la tabla detalle\_factura está vacía)

// Tiempo peor esperado: O(n) (Cuando la tabla detalle\_factura tiene n registros)

int inc = 0;

ResultSet rs;

rs = dfd.consultaBD("SELECT max(id\_detalle) as num FROM detalle\_factura;");

if (rs.next()) {

inc = rs.getInt(1) + 1;

} else {

inc = 1;

}

return inc;

}

**Metodo insertar\_detalle**

public void insertar\_detalle() throws SQLException {

// Tiempo mejor esperado: O(1) (Suponiendo que la inserción es rápida)

// Tiempo peor esperado: O(1) (Suponiendo que la inserción es rápida)

try {

String cadena = "insert into factura values ('" + incremento\_detalle() + "','" + getId\_factura() + "','" + getId\_producto() + "','" + "','" + getCantidad() + "','" + getPvp() + "','" + getPrecio\_uni() + ")";

dfd.actualizarBD(cadena);

} catch (SQLException e) {

}

}

**Metodo eliminar\_detalle**

public void eliminar\_detalle(String idDetalle) throws SQLException {

// Tiempo mejor esperado: O(1) (Suponiendo que la eliminación es rápida)

// Tiempo peor esperado: O(n) (Cuando la tabla detalle\_factura tiene n registros)

String sql = "DELETE FROM detalle\_factura WHERE id\_detalle = " + idDetalle;

dfd.actualizarBD(sql);

}

**Metodo consultaTabla**

public ResultSet consultaTabla(String sql) throws SQLException {

// Tiempo mejor esperado: O(1) (Suponiendo que la consulta es rápida)

// Tiempo peor esperado: Dependiendo de la complejidad de la consulta SQL

return dfd.consultaBD(sql);

}

**Clase Empleado\_beans**

**Metodo incremento\_empleado**

public int incremento\_empleado() throws SQLException {

// Tiempo mejor esperado: O(1) (Cuando la tabla empleado está vacía)

// Tiempo peor esperado: O(n) (Cuando la tabla empleado tiene n registros)

int inc = 0;

ResultSet rs;

rs = ed.consultaBD("SELECT max(id\_empleado) as num FROM empleado;");

if (rs.next()) {

inc = rs.getInt(1) + 1;

} else {

inc = 1;

}

return inc;

}

**Metodo insertar\_empleado**

public void insertar\_empleado() throws SQLException {

// Tiempo mejor esperado: O(1) (Suponiendo que la inserción es rápida)

// Tiempo peor esperado: O(1) (Suponiendo que la inserción es rápida)

try {

String cadena = "insert into empleado values ('" + incremento\_empleado() + "','" + getNombres() + "','" + getApellidos() + "','" + getCorreo() + "', '" + getTelefono() + "')";

ed.actualizarBD(cadena);

} catch (SQLException e) {

}

}

**Metodo consultaTabla**

public ResultSet consultaTabla(String sql) throws SQLException {

// Tiempo mejor esperado: O(1) (Suponiendo que la consulta es rápida)

// Tiempo peor esperado: Dependiendo de la complejidad de la consulta SQL

return ed.consultaBD(sql);

}

**Metodo eliminar\_empleado**

public void eliminar\_empleado(String idEmpleado) throws SQLException {

// Tiempo mejor esperado: O(1) (Suponiendo que la eliminación es rápida)

// Tiempo peor esperado: O(n) (Cuando la tabla empleado tiene n registros)

String sql = "DELETE FROM empleado WHERE id\_empleado = " + idEmpleado;

ed.actualizarBD(sql);

}

**Metodo consultar\_empleado**

public void consultar\_empleado() throws SQLException {

// Tiempo mejor esperado: O(1) (Cuando no hay registros en la tabla empleado)

// Tiempo peor esperado: O(n) (Cuando la tabla empleado tiene n registros)

ResultSet rs;

rs = ed.consultaBD("select \* from empleado");

while (rs.next()) {

System.out.print(rs.getInt(1) + " ");

System.out.print(rs.getString(2) + " ");

System.out.print(rs.getString(3) + " ");

System.out.print(rs.getString(4) + " ");

System.out.print(rs.getString(5) + " ");

System.out.println("");

}

}

**Metodo consultarEmpleadoPorNombre**

public ResultSet consultarEmpleadoPorNombre(String nombreProducto) throws SQLException {

// Tiempo mejor esperado: O(1) (Suponiendo que la consulta es rápida)

// Tiempo peor esperado: Dependiendo de la complejidad de la consulta SQL y el

String sql = "select \* from empleado WHERE nombres AND apellidos = ?";

PreparedStatement statement = ed.prepareStatement(sql);

statement.setString(1, nombreProducto);

return statement.executeQuery();

}

**Clase Factura\_beans**

**Metodo incremento\_factura**

public int incremento\_factura() throws SQLException {

// Tiempo mejor esperado: O(1) (Cuando la tabla factura está vacía)

// Tiempo peor esperado: O(n) (Cuando la tabla factura tiene n registros)

int inc = 0;

ResultSet rs;

rs = fd.consultaBD("SELECT max(ID\_\_FACTURA) as num FROM factura;");

if (rs.next()) {

inc = rs.getInt(1) + 1;

} else {

inc = 1;

}

return inc;

}

**Metodo insertar\_factura**

public void insertar\_factura() throws SQLException {

// Tiempo mejor esperado: O(1) (Suponiendo que la inserción es rápida)

// Tiempo peor esperado: O(1) (Suponiendo que la inserción es rápida)

try {

String cadena = "insert into factura values ('" + incremento\_factura() + "','" + getId\_empleado() + "','" + getId\_cliente() + "','" + "','" + getFecha() + "','" + getSubtotal() + "','" + getIva() + "','" + getTotal() + "')";

fd.actualizarBD(cadena);

} catch (SQLException e) {

}

}

**Metodo eliminar\_factura**

public void eliminar\_factura(String idFactura) throws SQLException {

// Tiempo mejor esperado: O(1) (Suponiendo que la eliminación es rápida)

// Tiempo peor esperado: O(n) (Cuando la tabla factura tiene n registros)

String sql = "DELETE FROM factura WHERE id\_factura = " + idFactura;

fd.actualizarBD(sql);

}

**Metodo consultaTabla**

public ResultSet consultaTabla(String sql) throws SQLException {

// Tiempo mejor esperado: O(1) (Suponiendo que la consulta es rápida)

// Tiempo peor esperado: Dependiendo de la complejidad de la consulta SQL

return fd.consultaBD(sql);

}

**Metodo consultar\_factura**

public void consultar\_factura() throws SQLException {

// Tiempo mejor esperado: O(1) (Cuando no hay registros en la tabla factura)

// Tiempo peor esperado: O(n) (Cuando la tabla factura tiene n registros)

ResultSet rs;

rs = fd.consultaBD("select \* from factura");

while (rs.next()) {

System.out.print(rs.getInt(1) + " ");

System.out.print(rs.getInt(2) + " ");

System.out.print(rs.getInt(3) + " ");

System.out.print(rs.getInt(4) + " ");

System.out.print(rs.getInt(5) + " ");

System.out.print(rs.getFloat(6) + " ");

System.out.print(rs.getFloat(7) + " ");

System.out.println("");

}

}

**Clase Producto\_beans**

**Metodo incremento\_Producto**

public int incremento\_Producto() throws SQLException {

// Tiempo mejor esperado: O(1) (Cuando la tabla producto está vacía)

// Tiempo peor esperado: O(n) (Cuando la tabla producto tiene n registros)

int inc = 0;

ResultSet rs;

rs = pd.consultaBD("SELECT max(id\_producto) as num FROM producto;");

if (rs.next()) {

inc = rs.getInt(1) + 1;

} else {

inc = 1;

}

return inc;

}

**Metodo insertar\_producto**

public void insertar\_producto() throws SQLException {

// Tiempo mejor esperado: O(1) (Suponiendo que la inserción es rápida)

// Tiempo peor esperado: O(1) (Suponiendo que la inserción es rápida)

try {

String cadena = "insert into producto values ('" + incremento\_Producto() + "','" + getNombreProducto() + "','" + getDescripcion() + "','" + getStock() + "','" + getPvp() + "')";

pd.actualizarBD(cadena);

} catch (SQLException e) {

}

}

**Metodo eliminar\_producto**

public void eliminar\_producto(String idProducto) throws SQLException {

// Tiempo mejor esperado: O(1) (Suponiendo que la eliminación es rápida)

// Tiempo peor esperado: O(n) (Cuando la tabla producto tiene n registros)

String sql = "DELETE FROM producto WHERE id\_producto = " + idProducto;

pd.actualizarBD(sql);

}

**Metodo consultaTabla**

public ResultSet consultaTabla(String sql) throws SQLException {

// Tiempo mejor esperado: O(1) (Suponiendo que la consulta es rápida)

// Tiempo peor esperado: Dependiendo de la complejidad de la consulta SQL

return pd.consultaBD(sql);

}

**Metodo consultar\_producto**

public void consultar\_producto() throws SQLException {

// Tiempo mejor esperado: O(1) (Cuando no hay registros en la tabla producto)

// Tiempo peor esperado: O(n) (Cuando la tabla producto tiene n registros)

ResultSet rs;

rs = pd.consultaBD("select \* from producto");

while (rs.next()) {

System.out.print(rs.getInt(1) + " ");

System.out.print(rs.getString(2) + " ");

System.out.print(rs.getString(3) + " ");

System.out.print(rs.getInt(4) + " ");

System.out.print(rs.getInt(5) + " ");

System.out.println("");

}

}

public void establecerConexion(Connection conexion) {

this.pd = (accesobd) conexion;

}

**Metodo consultarProductos**

public ResultSet consultarProductos() throws SQLException {

// Tiempo mejor esperado: O(1) (Suponiendo que la consulta es rápida)

// Tiempo peor esperado: Dependiendo de la complejidad de la consulta SQL

String consulta = "SELECT \* FROM producto";

return pd.consultaBD(consulta);

}

**Clase accesobd**

**Metodo conectarBD**

public void conectarBD(){

// Tiempo mejor esperado: O(1)

// Tiempo peor esperado: O(1)

try {

Driver driver =(Driver) Class.forName("com.mysql.jdbc.Driver").newInstance();

String cadena= "jdbc:mysql://"+ getHost()+ "/" + getBd();

conexion=DriverManager.getConnection(cadena, getUser(), getPasswd());

System.out.println("Conexion Exitosa a la base de Datos... Siuuuuuu");

} catch(Exception e){

System.out.println("Error en la conexion de la base de datos... :/" + e.toString());

}

}

**Metodo actualizarBD**

public void actualizarBD(String sql){

// Tiempo mejor esperado: O(1) (Suponiendo que la transacción es rápida)

// Tiempo peor esperado: O(1) (Suponiendo que la transacción es rápida)

try {

Statement stm = conexion.createStatement();

stm.executeUpdate(sql);

System.out.println("Transaccion exitosa... siuuuuuu");

} catch(Exception e){

System.out.println("Error em la transaccion... Rollblack " );

}

}

**Metodo consultaBD**

public ResultSet consultaBD(String sql)throws SQLException{

// Tiempo mejor esperado: O(1) (Suponiendo que la consulta es rápida)

// Tiempo peor esperado: Dependiendo de la complejidad de la consulta SQL

ResultSet cursor;

Statement stm = conexion.createStatement();

cursor = stm.executeQuery(sql);

return cursor;

}

**Metodo cerrarBD**

public void cerrarBD()throws SQLException{

// Tiempo mejor esperado: O(1)

// Tiempo peor esperado: O(1)

if (conexion != null) {

conexion.close();

System.out.println("Conexión cerrada");

}

}