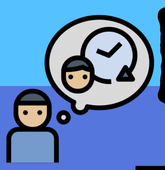
Programación 3

Refactorización de Software (back-end en Java)

Dr. Andrés Melgar



Recordando

Arquitectura de SW (estilo arquitectónico monolítico por capas)

Presentación

Front-End

Microsoft .NET

Lógica de Negocio

Java

BussinessObject

+regla_de_negocio()

Patrón DAO
(patrón de interacción con BD Data Mapper)

Dominio

Back-End

Model

-campo1
-campo2

Patrones:

- Data Transfer Object (diseño)
- Identity Field (persistencia)
- Sigle Table Inherence (persistencia)

Persistencia

DataAccessObject

+insertar()
+actualizar()
+eliminar()
+consultar()

DataAccessObjetImplementation

-conexion
-resultSet

+insertar()
+actualizar()
+eliminar()
+consultar()

Base de datos

MySQL

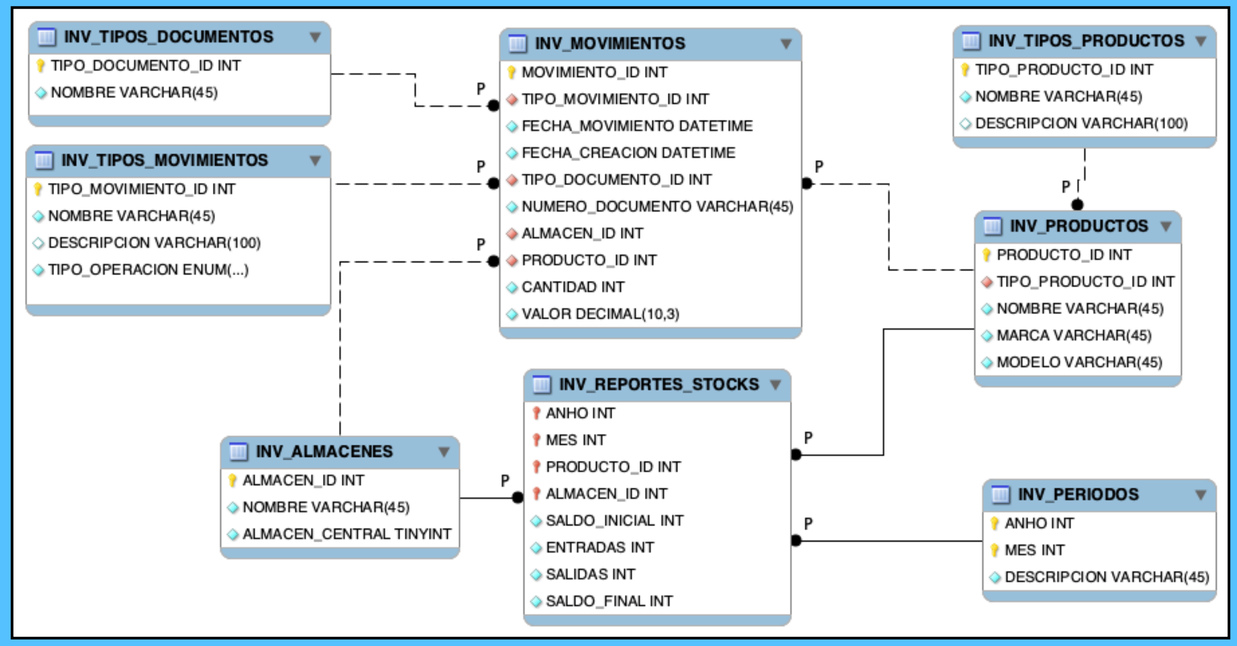
Microsoft® SQL Server®

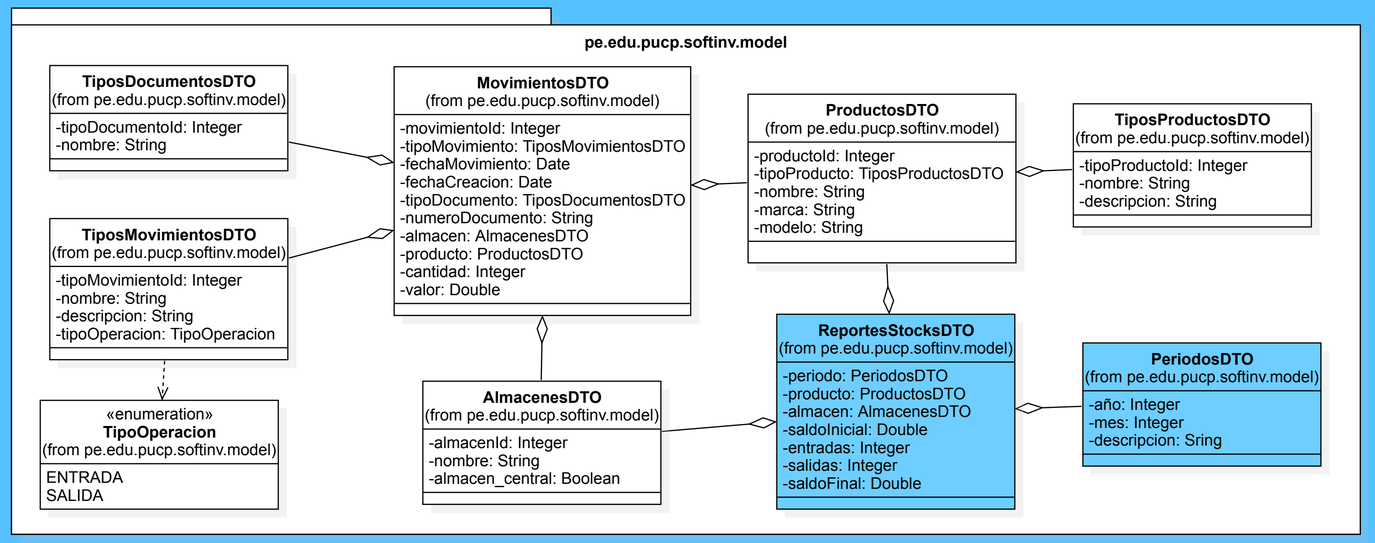
Tarea

Recordando

Trabajo para la casa

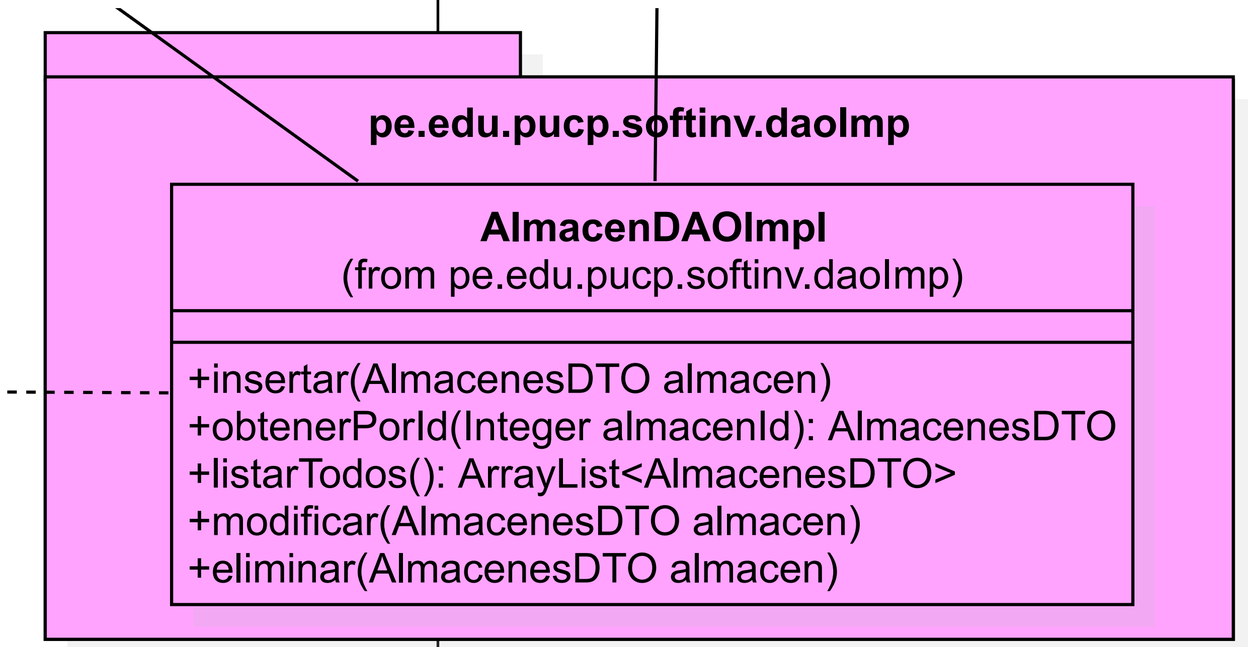
- Implementar todas las clases de la capa del dominio en el proyecto **SoftInvModel**.
- Para cada entidad del dominio, implementar todas las operaciones CRUD en el proyecto **SoftInvPersistance**, incluyendo las pruebas unitarias (menos ReportesStocksDTO).
- Para cada operación CRUD, implemente su invocación en el proyecto **SoftInvBusiness**, incluyendo las pruebas unitarias.



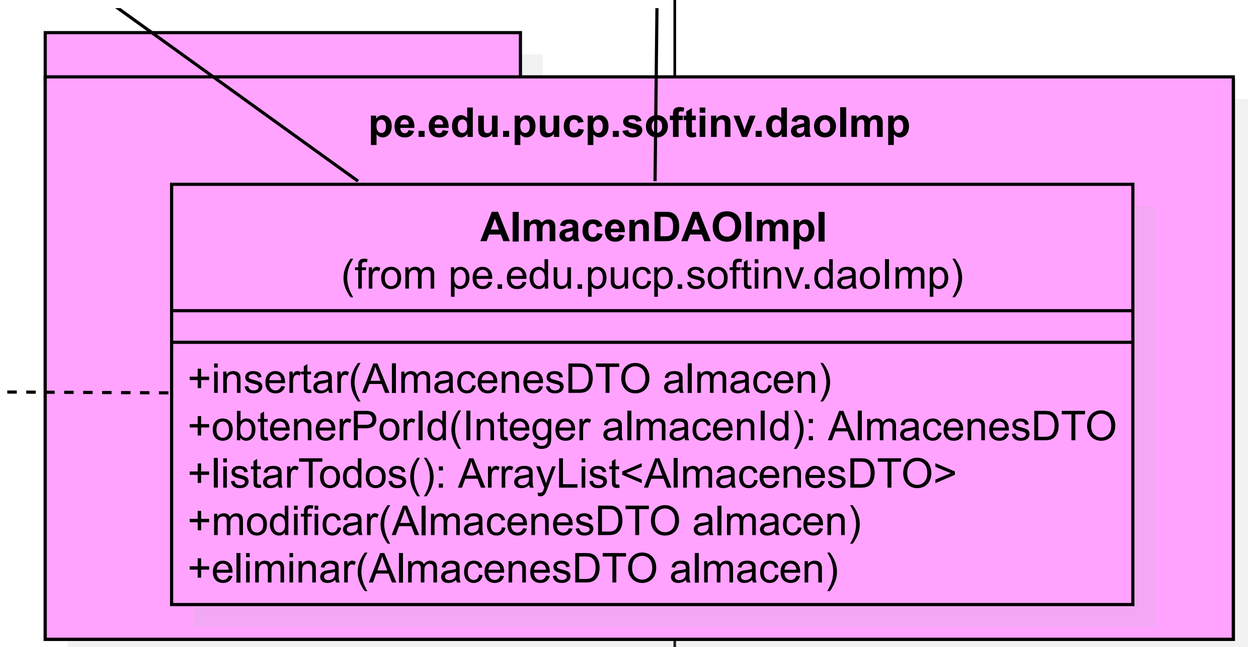


¿Qué problema han encontrado en la resolución de la tarea en términos de diseño de software?

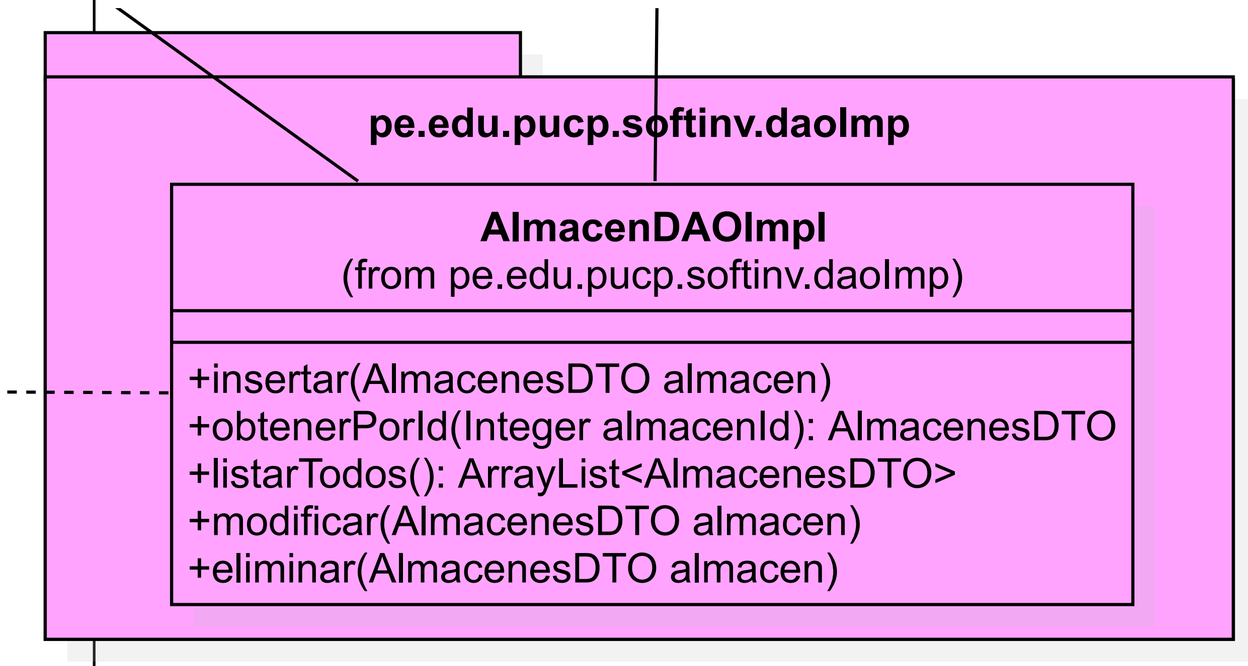
```java
public Integer insertar(AlmacenesDTO almacen) {
    int resultado = 0;
    try {
        this.conexion = DBManager.getInstance().getConnection();
        this.conexion.setAutoCommit(false);
        String sql = "INSERT INTO INV_ALMACENES (NOMBRE, ALMACEN_CENTRAL) VALUES (?,?)";
        this.statement = this.conexion.prepareCall(sql);
        this.statement.setString(1, almacen.getNombre());
        this.statement.setInt(2, almacen.getAlmacen_central() ? 1 : 0);
        this.statement.executeUpdate();
        resultado = this.retornarUltimoAutoGenerado();
        this.conexion.commit();
    } catch (SQLException ex) {
        System.err.println("Error al intentar insertar - " + ex);
        try {
            if (this.conexion != null) {
                this.conexion.rollback();
            }
        } catch (SQLException ex1) {
            System.err.println("Error al hacer rollback - " + ex1);
        }
    } finally {
        try {
            if (this.conexion != null) {
                this.conexion.close();
            }
        } catch (SQLException ex) {
            System.err.println("Error al cerrar la conexión - " + ex);
        }
    }
    return resultado;
}
```



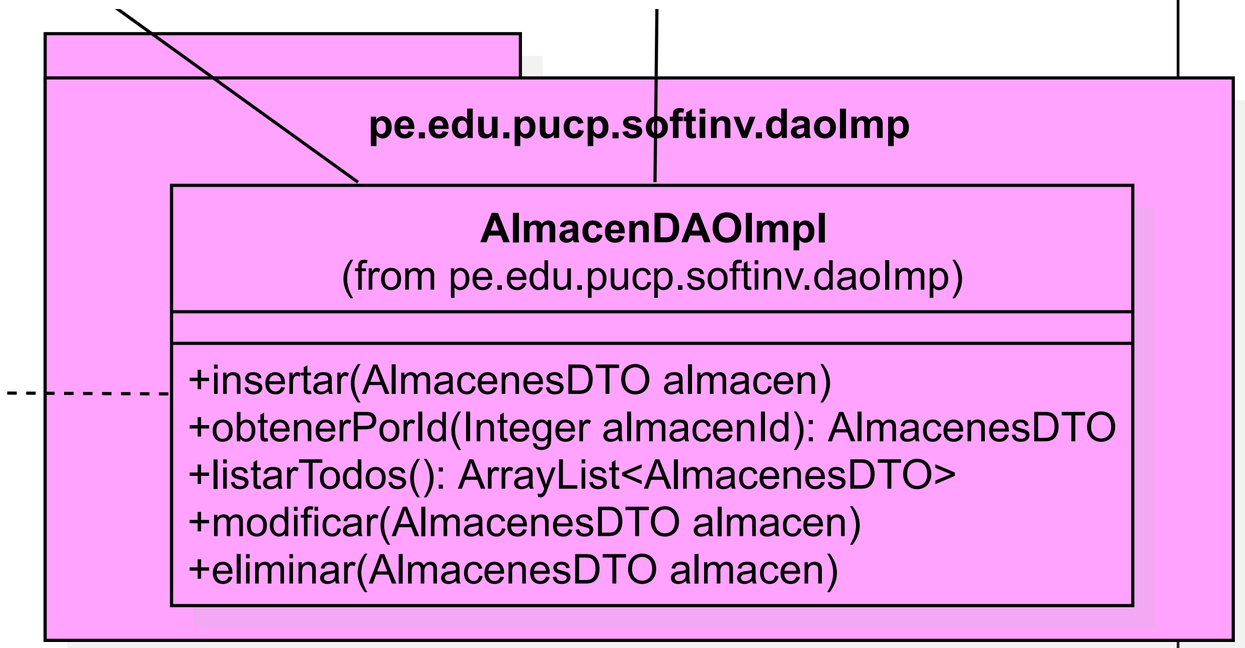
```java
public Integer modificar(AlmacenesDTO almacen) {
    int resultado = 0;
    try {
        this.conexion = DBManager.getInstance().getConnection();
        this.conexion.setAutoCommit(false);
        String sql = "UPDATE INV_ALMACENES SET NOMBRE=?, ALMACEN_CENTRAL=? WHERE ALMACEN_ID=?";
        this.statement = this.conexion.prepareCall(sql);
        this.statement.setString(1, almacen.getNombre());
        this.statement.setInt(2, almacen.getAlmacen_central() ? 1 : 0);
        this.statement.setInt(3, almacen.getAlmacenId());
        resultado = this.statement.executeUpdate();
        this.conexion.commit();
    } catch (SQLException ex) {
        System.err.println("Error al intentar modificar - " + ex);
        try {
            if (this.conexion != null) {
                this.conexion.rollback();
            }
        } catch (SQLException ex1) {
            System.err.println("Error al hacer rollback - " + ex1);
        }
    } finally {
        try {
            if (this.conexion != null) {
                this.conexion.close();
            }
        } catch (SQLException ex) {
            System.err.println("Error al cerrar la conexión - " + ex);
        }
    }
    return resultado;
}
```



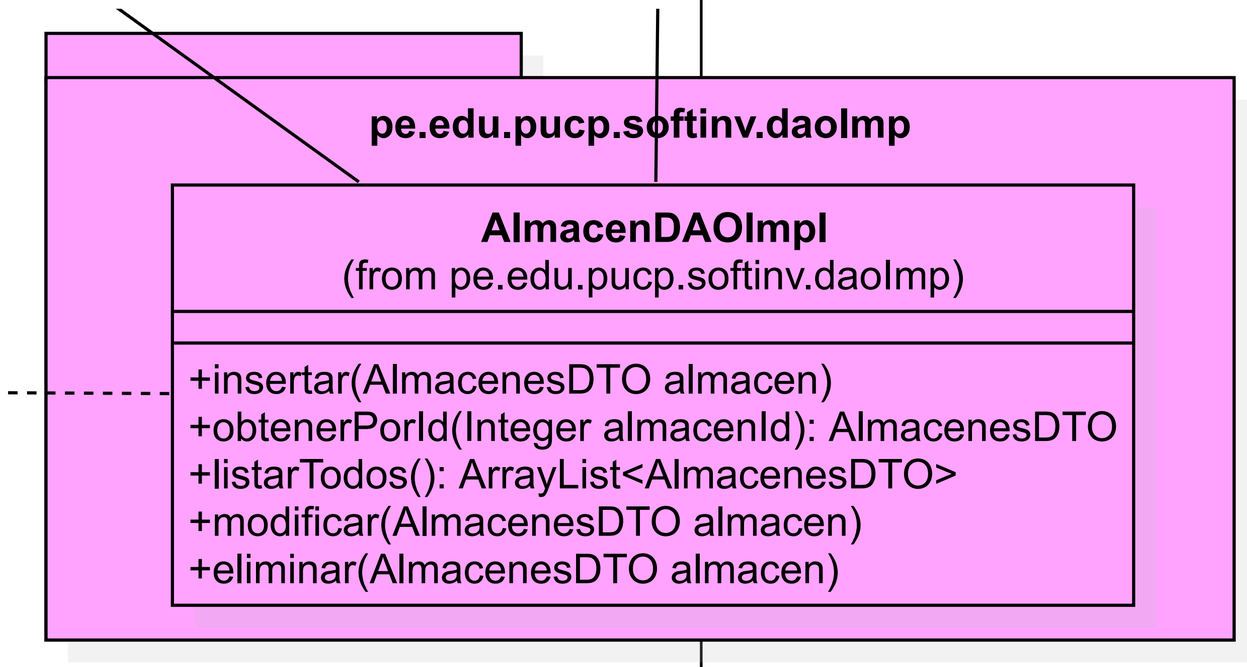
```java
public Integer eliminar(AlmacenesDTO almacen) {
        int resultado = 0;
        try {
            this.conexion = DBManager.getInstance().getConnection();
            this.conexion.setAutoCommit(false);
            String sql = "DELETE FROM INV_ALMACENES WHERE ALMACEN_ID=?";
            this.statement = this.conexion.prepareCall(sql);
            this.statement.setInt(1, almacen.getAlmacenId());
            resultado = this.statement.executeUpdate();
            this.conexion.commit();
        } catch (SQLException ex) {
            System.err.println("Error al intentar eliminar - " + ex);
            try {
                if (this.conexion != null) {
                    this.conexion.rollback();
                }
            } catch (SQLException ex1) {
                System.err.println("Error al hacer rollback - " + ex1);
            }
        } finally {
            try {
                if (this.conexion != null) {
                    this.conexion.close();
                }
            } catch (SQLException ex) {
                System.err.println("Error al cerrar la conexión - " + ex);
            }
        }
        return resultado;
    }
```



```java
public AlmacenesDTO obtenerPorId(Integer almacenId) {
        AlmacenesDTO almacen = null;
        try {
            this.conexion = DBManager.getInstance().getConnection();
            String sql = "SELECT ALMACEN_ID, NOMBRE, ALMACEN_CENTRAL FROM INV_ALMACENES WHERE ALMACEN_ID = ?";
            this.statement = this.conexion.prepareCall(sql);
            this.statement.setInt(1, almacenId);
            this.resultSet = this.statement.executeQuery();
            if (this.resultSet.next()) {
                almacen = new AlmacenesDTO();
                almacen.setAlmacenId(this.resultSet.getInt("ALMACEN_ID"));
                almacen.setNombre(this.resultSet.getString("NOMBRE"));
                almacen.setAlmacen_central(this.resultSet.getInt("ALMACEN_CENTRAL") == 1);
            }
        } catch (SQLException ex) {
            System.err.println("Error al intentar obtenerPorId - " + ex);
        } finally {
            try {
                if (this.conexion != null) {
                    this.conexion.close();
                }
            } catch (SQLException ex) {
                System.err.println("Error al cerrar la conexión - " + ex);
            }
        }
        return almacen;
    }
```



```java
public ArrayList<AlmacenesDTO> listarTodos(){
    ArrayList<AlmacenesDTO> listaAlmacenes = new ArrayList<>();
    try {
        this.conexion = DBManager.getInstance().getConnection();
        String sql = "SELECT ALMACEN_ID, NOMBRE, ALMACEN_CENTRAL FROM INV_ALMACENES";
        this.statement = this.conexion.prepareCall(sql);
        this.resultSet = this.statement.executeQuery();
        while (this.resultSet.next()) {
            AlmacenesDTO almacen = new AlmacenesDTO();
            almacen.setAlmacenId(this.resultSet.getInt("ALMACEN_ID"));
            almacen.setNombre(this.resultSet.getString("NOMBRE"));
            almacen.setAlmacen_central(this.resultSet.getInt("ALMACEN_CENTRAL") == 1);
            listaAlmacenes.add(almacen);
        }
    } catch (SQLException ex) {
        System.err.println("Error al intentar listarTodos - " + ex);
    } finally {
        try {
            if (this.conexion != null) {
                this.conexion.close();
            }
        } catch (SQLException ex) {
            System.err.println("Error al cerrar la conexión - " + ex);
        }
    }
    return listaAlmacenes;
}
```



AlmacenDAOImpl

```
public Integer insertar(AlmacenesDTO almacen) {
        int resultado = 0;
        try {
            this.conexion = DBManager.getInstance().getConnection();
            this.conexion.setAutoCommit(false);
            String sql = "INSERT INTO INV_ALMACENES (NOMBRE,
ALMACEN_CENTRAL) VALUES (?,?)";
            this.statement = this.conexion.prepareCall(sql);
            this.statement.setString(1, almacen.getNombre());
            this.statement.setInt(2, almacen.getAlmacen_central() ? 1 : 0);
            this.statement.executeUpdate();
            resultado = this.retornarUltimoAutoGenerado();
            this.conexion.commit();
        } catch (SQLException ex) {
            System.err.println("Error al intentar insertar - " + ex);
            try {
                if (this.conexion != null) {
                    this.conexion.rollback();
                }
            } catch (SQLException ex1) {
                System.err.println("Error al hacer rollback - " + ex1);
            }
        } finally {
            try {
                if (this.conexion != null) {
                    this.conexion.close();
                }
            } catch (SQLException ex) {
                System.err.println("Error al cerrar la conexión - " + ex);
            }
        }
        return resultado;
    }
```

TipoDocumentoDAOImpl

```
public Integer insertar(TiposDocumentosDTO tipoDocumento) {
        int resultado = 0;
        try {
            this.conexion = DBManager.getInstance().getConnection();
            this.conexion.setAutoCommit(false);
            String sql = "INSERT INTO INV_TIPOS_DOCUMENTOS
(TIPO_DOCUMENTO_ID, NOMBRE) VALUES (?,?)";
            this.statement = this.conexion.prepareCall(sql);
            this.statement.setInt(1, tipoDocumento.getTipoDocumentoId());
            this.statement.setString(2, tipoDocumento.getNombre());
            resultado = this.statement.executeUpdate();
            this.conexion.commit();
        } catch (SQLException ex) {
            System.err.println("Error al intentar insertar - " + ex);
            try {
                if (this.conexion != null) {
                    this.conexion.rollback();
                }
            } catch (SQLException ex1) {
                System.err.println("Error al hacer rollback - " + ex1);
            }
        } finally {
            try {
                if (this.conexion != null) {
                    this.conexion.close();
                }
            } catch (SQLException ex) {
                System.err.println("Error al cerrar la conexión - " + ex);
            }
        }
        return resultado;
    }
```

¿Qué debemos hacer?

Refactorización de Software

Refactorización de Software

Definición

- La refactorización de software es el proceso de **mejorar el código** fuente sin cambiar su comportamiento externo.
- Su objetivo es hacer que el código sea **más limpio, legible y mantenible**, reduciendo la complejidad y mejorando su estructura.
- **No cambia la funcionalidad** del programa, solo mejora su estructura.

Refactorización de Software

Ventajas

- Facilita el **mantenimiento** y evolución del software.
- Reduce la **duplicación** de código.
- Mejora la **legibilidad** y la organización del código.
- Ayuda a encontrar y corregir **errores** ocultos.
- Optimiza el **rendimiento** en algunos casos.

Refactorización de Software

Refactorización usando el Paradigma Orienta a Objetos

- **Extract Method**
  - Problema: Un método es muy largo o realiza múltiples tareas.
    Solución: Dividirlo en métodos más pequeños y reutilizables.
- **Reemplazar Código Duplicado con Herencia o Composición**
  - Problema: Código repetido en múltiples clases.
  - Solución: Usar herencia (si hay relación “es un”) o composición (si hay relación “tiene un”).

Refactorización de Software

Refactorización usando el Paradigma Orienta a Objetos

- **Convertir Código Procedural a Orientado a Objetos**
  - Problema: Código estructurado sin encapsulación ni reutilización.
  - Solución: Convertir en clases con métodos adecuados.
- **Sustituir Condiciones Complejas con Polimorfismo**
  - Problema: Uso excesivo de if-else o switch con lógica difícil de entender.
  - Solución: Aplicar polimorfismo para delegar la responsabilidad.

Refactorización de Software

Refactorización usando el Paradigma Orienta a Objetos

- **Introducir Interfaces para Reducir Acoplamiento**
  - Problema: Una clase depende de una implementación específica.
  - Solución: Usar interfaces para que dependa de una abstracción.
- **Reemplazar Herencia con Composición**
  - Problema: La herencia a veces causa problemas de acoplamiento fuerte.
  - Solución: Usar composición cuando la relación no es “es un”.

Recomendación del profesor

- Leer algún libro de:
  - **Código limpio** (*clean code*).
  - **Refactorización** de software (*software refactoring*).
  - Desarrollo guiado por pruebas (**TDD** por sus siglas en inglés de *“Test Driven Development”*)