**Brief Overview of Packages Used & General Flow of the Program**

**1. Packages Used**

The program makes use of several standard Java packages:

| **Package** | **Purpose** |
| --- | --- |
| org.w3c.dom | Used for parsing XML/XSD documents (DOM API). |
| javax.xml.parsers | Provides XML parsers for DOM-based XML processing. |
| javax.xml.validation | Used for validating XML against an XSD schema. |
| java.sql | Provides database connectivity using JDBC. |
| java.util | Used for handling collections like List, ArrayList, and HashMap. |

**2. General Flow of the Program**

1. **Verify meta-model.xsd against meta-meta-model.xsd**
   * Ensures that the meta-model.xsd is correctly defined before proceeding.
   * If validation fails, the program stops execution.
2. **Generate SQL Table Definitions from meta-model.xsd**
   * Parses the XSD and dynamically constructs CREATE TABLE SQL statements.
   * Includes attributes as columns and foreign key relationships.
3. **Create Tables in the Database**
   * Executes the generated SQL queries to create the necessary database tables.
4. **Parse XML Data from store.xml**
   * Reads the XML file and converts it into Java objects (Category, Subcategory, Product).
   * Maintains relationships between the parsed elements.
5. **Insert Data into the Database**
   * Uses the parsed data to populate the dynamically created tables while maintaining foreign key constraints.

**3. Function Overview (Key Methods)**

**1. Main.java**

Handles the overall execution flow.

public static void main(String[] args)

* **Step 1:** Validates meta-model.xsd against meta-meta-model.xsd.
* **Step 2:** Generates table creation SQL from meta-model.xsd.
* **Step 3:** Creates the tables in the database.
* **Step 4:** Parses store.xml to extract category, subcategory, and product data.
* **Step 5:** Inserts parsed data into the database.

**2. XSDValidator.java**

Validates meta-model.xsd against meta-meta-model.xsd.

public static boolean validateXSD(String xsdPath, String xmlPath)

* Uses SchemaFactory, Schema, and Validator to check if meta-model.xsd conforms to meta-meta-model.xsd.

**3. XSDToSQLConverter.java**

Converts the XSD structure into SQL CREATE TABLE statements.

public static List<String> generateTablesFromXSD(String xsdFilePath)

* Parses meta-model.xsd using org.w3c.dom API.
* Reads elements (Entity) and attributes.
* Constructs SQL queries with primary keys and foreign keys.
* Returns a list of SQL CREATE TABLE statements.

**4. DatabaseConnector.java**

Handles database interactions.

jpublic static void executeTableCreationSQL(List<String> tableDefinitions)

* Connects to the database.
* Executes the provided CREATE TABLE statements.

public static int executeInsertAndGetGeneratedKey(String sql, String[] generatedColumns, Object[] parameters)

* Inserts records dynamically and retrieves the generated primary key.

**5. XMLParser.java**

Parses store.xml to extract structured data.

java

CopyEdit

public static List<Category> parseXMLAndGenerateData(String xmlFilePath)

* Reads store.xml using DocumentBuilderFactory.
* Extracts Category, Subcategory, and Product objects.

**Final Summary**

1️ **Validation:** Check if meta-model.xsd is valid.  
2️ **Table Generation:** Create SQL tables dynamically from XSD.  
3️ **Database Setup:** Execute SQL statements to create tables.  
4️ **XML Parsing:** Convert store.xml into Java objects.  
5️ **Data Insertion:** Populate database while maintaining relationships.

**Dependencies JAXB for XML parser, mysql, postgresql drivers**

**Also check sql commands used to create database and user**