Use Apache XMLBeans tool and JAXB to manipulate XML and Java Objects.

Purpose:

1. We want to infer a XML schema from an existing XML document.
2. We want to programmatically manipulate the "DOM" objects in the XML so that we can programmatically modify, add, and delete information in the XML.

Possible solution:

1. We can use a utility to infer an XML schema from a XML file.
2. Then we use another utility to generate programmable classes/objects from the inferred XML schema.
3. We can then manipulate those classes/objects in computer memory.
4. Finally, we use a utility to convert the manipulated objects back into an XML file.

Implementation Steps:

Go to your project root directory:

cd C:\Projects\XML_Project

The directory contains a Tool directory where XMLBeans utility resides and a source_xml directory where the original source XML file, cd_catalog.xml, resides.

Generate XML schema from XML file by inst2xsd (instance to xsd) of XMLbeans package:

.\Tool\xmlbeans-5.0.3\bin\inst2xsd .\source_xml\cd_catalog.xml -outDir .\generated_xml -outPrefix cd_catalog

The above command generates cd_catalog0.xsd file from cd_catalog.xml

Use Java JAXB library tool xjc to generate Java source code from XML schema (XSD):

C:\Java\jdk1.8.0_202\bin\xjc -d .\src -p org.dragon.yunpeng.generated.xjc -b .\generated_xml\bindings.xml .\generated_xml\cd_catalog0.xsd

Note: Java needs to be installed first. JAXB should come with Java.

-d: destination directory where generated source code will be in

-p: application package, can name anything as your java application package

-b: binding XML file where generated class names can be specified. If binding XML is not provided, all generated Java class names will have “Type” suffix.

The above command generates 3 files from cd_catalog0.xsd:

1. Catalog.java
2. CD.java
3. ObjectFactory.java

Use Eclipse IDE to do Java coding:

Create a Java Project in Eclipse.

Choose src as the source folder.

Modify Category.java:

In Category.java, locate @XmlType(name = "CATALOGType"...), change "CATALOGType" to "CATALOG".

Add root annotation @XmlRootElement(name = "CATALOG") to Catalog.java.

Add constructor, toString(), and addACD() methonds:

```
public Catalog() { this.cd = new ArrayList(); }

@Override
public String toString() {

		StringBuilder str = new StringBuilder("[");
		for (CD aCD : cd) {
			str.append(aCD);
			str.append(" | ");
		}

		str.append("]");

		return str.toString();
}

public void addACD(CD aCD) {
		this.cd.add(aCD);
}
```

Modify CD.java:

In CD.java, locate @XmlType(name = "CDType"...), change "CDType" to "CD".

Add toString() method for debug printing:

```
@Override

public String toString() {

        return "title: " + this.title + ", artist: " + this.artist + ", company: " + this.company + ", country: " +
        this.country + ", price: " + this.price + ", year: " + this.year;

}
```

Create a main java class (See MainApplication.java): Implement methods that use JAXB unmarshal (XML to Java objects) and marshal (Java objects to XML).

```
public static void unmarshallTest() {

        try {

                JAXBContext jaxbContext = JAXBContext.newInstance(Catalog.class);

                Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();

                // Unmarshalling (Convert XML instance into Java objects)

                Catalog catalog = (Catalog) jaxbUnmarshaller.unmarshal(new
                File("C:\Projects\XML_Project\source_xml\cd_catalog.xml"));

           System.out.println(catalog);

        } catch (JAXBException e) {

           e.printStackTrace();
        }
}

public static void marshallTest() {

        //Create a Catalog object
        Catalog catalog = new Catalog();

        //Create a CD object
        CD cd = new CD();
        cd.setARTIST("Yunpeng Li");
        cd.setCOMPANY("BAE");
        cd.setCOUNTRY("USA");
        cd.setPRICE(15);
        cd.setTITLE("Enigma");
```

```
        cd.setYEAR(Short.valueOf("2021"));

        //Add the new CD object into the Catalog object
        catalog.addACD(cd);

        JAXBContext jaxbContext;

        try {

           jaxbContext = JAXBContext.newInstance(Catalog.class);

           Marshaller jaxbMarshaller = jaxbContext.createMarshaller();
           File file = new File("C:\Projects\XML_Project\generated_xml\cd_catalog_new.xml");

        // output pretty printed

        jaxbMarshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);

           // Marshalling (Convert Java objects into XML)
           jaxbMarshaller.marshal(catalog, file);
           jaxbMarshaller.marshal(catalog, System.out);
        } catch (JAXBException e) {
           e.printStackTrace();
        }
}
```

Now we can programmatically load existing XML file and create Java object representation from it, then we can add or modify the values in Java objects and marshal back to XML format.

To run MainApplication.java in Eclipse:

TODO: Use Java GUI to manipulate CD objects in Catalog.

Better alternative that uses salami slice design (No “Type” suffix is generated):

.\Tool\xmlbeans-5.0.3\bin\inst2xsd -design ss -enumerations never .\source_xml\cd_catalog.xml -outDir .\generated_xml -outPrefix cd_catalog_ss

C:\Java\jdk1.8.0_202\bin\xjc -d .\src -p org.dragon.yunpeng.generated.xjc2 .\generated_xml\cd_catalog_ss0.xsd

Useful Links:

Apache XMLBeans: https://xmlbeans.apache.org/

JAXB: Generate Classes from XSD: https://examples.javacodegeeks.com/core-java/xml/bind/jaxb-generate-classes-xsd/

Marshalling and Unmarshalling in JAXB: https://dzone.com/articles/introduction-to-jaxb-20

Eclipse Download: https://www.eclipse.org/