**Методические указания к выполнению лабораторных работ по дисциплине «Объектно-ориентированные базы данных»**

**Лабораторная работа 1**

Тема: Разработка объектно-ориентированной модели данных прикладной задачи.

Цель: Научиться разрабатывать объектно-ориентированную модель данных прикладной задачи.

Рассмотрим небольшой фрагмент задачи по ведению счета клиента банка: необходимо вести реестр клиентов, банковских счетов клиента (у каждого может быть несколько), сотрудников (проводящих операции по счету).

Выделим основные сущности:

Банк

Сотрудник

Счет

Клиент

Операция по счету

Классы доступны по ссылке: <https://github.com/ksenikeev/oodb>

**Лабораторная работа 2**

**Лабораторная работа 3**

Тема: XML формат. XSD схема. Хранение и обработка объектов с использованием XML формата.

XML (англ. eXtensible Markup Language) — расширяемый язык разметки, предназначенный для хранения и передачи данных.

С помощью XML описываются данные в виде структурированного документа. Документ состоит из элементов, каждый из которых начинается с открывающего тега и завершается закрывающим тегом

<tag\_name>**value**</tag\_name>

Между тегами указывается значение элемента.

Эемент может содержать атрибуты, которые перечисляются внутри открывающего тега:

<tag\_name atribute1="a\_value1" atribute2="a\_value1">**value**</tag\_name>

Значением элемента может выступать вложенный элемент.

<tag1>

<tag2>**value**</tag2>

</tag2>

Таким образом структура XML документа иерархическая, во главе иерархии находится корневой элемент.

Пример XML документа, описывающего группу института:

<?xml version="1.0" encoding="UTF-8"?>

<institut>

<name>**ИВМиИТ**</name>

<groupp>

<name>09-751</name>

<year>2017</year>

<students>

<student>Bulat</student>

<student>Karina</student>

<student>Marat</student>

</students>

</groupp>

</institut>

Можно и так:

<?xml version="1.0" encoding="UTF-8"?>

<institut name="ИВМиИТ">

<groupp name="09-751" year="2017">

<students>

<student name="Bulat"/>

<student name="Karina"/>

<student name="Marat"/>

</students>

</groupp>

</institut>

Документы при этом не эквивалентны.

**Инструменты работы с XML**

Мы будем рассматривать JAXB (Java Architecture for XML Binding), технологию, позволяющую преобразовывать объект Java в XML документ, и обратно XML документ в объект Java.

В первом случае преобразование именуют «marshalling».

Обратное преобразование - «unmarshalling».

Преобразование объекта будем рассматривать на следующей структуре классов (в дополнение к описанию полей указаны аннотации, управляющие процессом преобразования):

**Bank.java**

**package** ru.icmit.oodb.lab3.model;

**import** javax.xml.bind.annotation.XmlElement;

**import** javax.xml.bind.annotation.XmlElementWrapper;

**import** javax.xml.bind.annotation.XmlRootElement;

**import** java.util.List;

@XmlRootElement(name = **"bank"**)

**public class** Bank {

**private** String **name**;

**private** List<Client> **clients**;

**private** List<Transaction> **transactions**;

**private** List<BankAccount> **bankAccounts**;

**public** String getName() {

**return name**;

}

**public void** setName(String name) {

**this**.**name** = name;

}

@XmlElementWrapper(name = **"clients"**)

@XmlElement(name = **"client"**)

**public** List<Client> getClients() {

**return clients**;

}

**public void** setClients(List<Client> clients) {

**this**.**clients** = clients;

}

@XmlElementWrapper(name = **"transactions"**)

@XmlElement(name = **"transaction"**)

**public** List<Transaction> getTransactions() {

**return transactions**;

}

**public void** setTransactions(List<Transaction> transactions) {

**this**.**transactions** = transactions;

}

@XmlElementWrapper(name = **"bankAccounts"**)

@XmlElement(name = **"bankAccount"**)

**public** List<BankAccount> getBankAccounts() {

**return bankAccounts**;

}

**public void** setBankAccounts(List<BankAccount> bankAccounts) {

**this**.**bankAccounts** = bankAccounts;

}

@Override

**public** String toString() {

**return "Bank: "** +

**"name='"** + **name** + **'\n'** +

**", clients="** + **clients** + **'\n'** +

**", transactions="** + **transactions** + **'\n'** +

**", bankAccounts="** + **bankAccounts**;

}

}

**Client.java**

**package** ru.icmit.oodb.lab3.model;

**import** javax.xml.bind.annotation.XmlElement;

**import** javax.xml.bind.annotation.XmlElementWrapper;

**import** java.util.List;

**public class** Client **extends** Person {

**public** Client(String firstName, String lastName, String phoneNumber, String email) {

**super**(firstName, lastName, phoneNumber, email);

}

**public** Client() {

}

**private** List<BankAccount> **accounts**;

@XmlElementWrapper(name = **"accounts"**)

@XmlElement(name = **"account"**)

**public** List<BankAccount> getAccounts() {

**return accounts**;

}

**public void** setAccounts(List<BankAccount> accounts) {

**this**.**accounts** = accounts;

}

@Override

**public** String toString() {

**return "Client: "** + getLastName() +

**", accounts="** + **accounts** +

**'\n'**;

}

}

**Person.java**

**package** ru.icmit.oodb.lab3.model;

**public class** Person {

**private** String **firstName**;

**private** String **lastName**;

**private** String **phoneNumber**;

**private** String **email**;

...

}

**Transaction.java**

**package** ru.icmit.oodb.lab3.model;

**import** java.util.Date;

**public class** Transaction {

**private** BankAccount **bancAccount**;

**private** Date **transactionDate**;

**private** Client **client**;

**private double amount**;

...

}

**BankAccount.java**

**package** ru.icmit.oodb.lab3.model;

**import** javax.xml.bind.annotation.XmlElement;

**public class** BankAccount {

**private long accountNumber**;

**private double balance**;

**public** BankAccount(**long** accountNumber) {

**this**.**accountNumber** = accountNumber;

**this**.**balance** = 0;

}

**public** BankAccount() {

}

@XmlElement(name = **"accountNumber"**)

**long** getAccountNumber() {

**return accountNumber**;

}

@XmlElement(name = **"balance"**)

**double** getBalance() {

**return balance**;

}

**public void** setAccountNumber(**long** accountNumber) {

**this**.**accountNumber** = accountNumber;

}

**public void** setBalance(**double** balance) {

**this**.**balance** = balance;

}

**protected void** increaseBalance(**double** cash) {

**this**.**balance** += cash;

}

**protected boolean** reduceBalance(**double** cash) {

**if** (**balance** < cash) {

**return false**;

} **else** {

**this**.**balance** -= cash;

**return true**;

}

}

@Override

**public** String toString() {

**return "BankAccount:"** +

**"accountNumber="** + **accountNumber** +

**", balance="** + **balance** +

**'\n'**;

}

}

Перечисленные выше классы описывают предметную область «Банк». Основной класс Bank содержит список клиентов (List<Client> **clients**), список расчетных счетов (List<BankAccount> **bankAccounts)**, список транзакций (List<Transaction> **transactions )**.

**Аннотации JAXB**

Наша задача — сохранить объект-Банк в XML файл bank.xml

Для указания корневого элемента XML документа мы используем аннотацию

@XmlRootElement(name = **"bank"**)

над определением класса Bank.

Для управляемого формирования элементов XML документа используют аннотации

@XmlElement(name = **"…"**), проставляемые над методом get...() соответствующего полю класса. Если аннотация не используется, то в качестве имени элемента будет использовано имя поля класса.

Списки удобно преобразовывать, используя элемент-обертку над набором одноименных элементов списка. В нашем случае мы хотим преобразовать в XML элемент поле List<Client> **clients**.

<clients>

<client>...</client>

<client>...</client>

...

<client>...</client>

</clients>

Такая структура формируется с помощью аннотаций:

@XmlElementWrapper(name = **"clients"**)

@XmlElement(name = **"client"**)

**public** List<Client> getClients() {

**return clients**;

}

Далее рассмотрим методы работы JAXB с подготовленными к преобразованию классами.

Основные методы преобразования описаны в классе Lab3XMLService

**package** ru.icmit.oodb.lab3;

**import** ru.icmit.oodb.lab3.model.Bank;

**import** javax.xml.bind.JAXBContext;

**import** javax.xml.bind.JAXBException;

**import** javax.xml.bind.Marshaller;

**import** javax.xml.bind.Unmarshaller;

**import** java.io.File;

**public class** Lab3XMLService {

*/\*\**

*\* метод saveBankData сохраняет объект типа Bank в XML документ*

*\*/*

**public static void** saveBankData(Bank bank) {

**try** {

JAXBContext context = JAXBContext.*newInstance*(Bank.**class**);

Marshaller marshaller = context.createMarshaller();

*// устанавливаем флаг для читабельного вывода XML в JAXB*

marshaller.setProperty(Marshaller.***JAXB\_FORMATTED\_OUTPUT***, Boolean.***TRUE***);

*// маршаллинг объекта в файл*

marshaller.marshal(bank, **new** File(**"bank.xml"**));

} **catch** (JAXBException e) {

e.printStackTrace();

}

}

*/\*\**

*\* Метод loadBankFromXML преобразует XML документ в объект типа Bank*

*\*/*

**public static** Bank loadBankFromXML() {

**try** {

*// создаем объект JAXBContext - точку входа для JAXB*

JAXBContext jaxbContext = JAXBContext.*newInstance*(Bank.**class**);

Unmarshaller un = jaxbContext.createUnmarshaller();

**return** (Bank) un.unmarshal(**new** File(**"bank.xml"**));

} **catch** (JAXBException e) {

e.printStackTrace();

}

**return null**;

}

}

**Следующий пример демонстрирует сохранения объекта в XML документ:**

**package** ru.icmit.oodb.lab3;

**import** ru.icmit.oodb.lab3.model.Bank;

**import** ru.icmit.oodb.lab3.model.BankAccount;

**import** ru.icmit.oodb.lab3.model.Client;

**import** java.util.ArrayList;

**import** java.util.List;

*/\*\**

*\* Пример, демонстрирующий сохранение (marshalling) объекта в XML документ*

*\*/*

**public class** Lab3MainSave {

**public static void** main(String[] args) {

*// Создаем объект банк*

Bank bank = **new** Bank();

*// Создаем объекты - клиенты банка*

Client client1 = **new** Client(**"Иван"**,**"Иванов"**,**""**,**""**);

Client client2 = **new** Client(**"Петр"**,**"Петров"**,**""**,**""**);

*// Добавляем объекту client1 расчетный счет*

BankAccount account = **new** BankAccount(1299876);

List<BankAccount> accounts = **new** ArrayList<>();

accounts.add(account);

client1.setAccounts(accounts);

List<Client> clients = **new** ArrayList<>();

clients.add(client1);

clients.add(client2);

*// Прикрепляем клиентов к банку*

bank.setClients(clients);

*// сохраняем объект в в XML документ*

Lab3XMLService.*saveBankData*(bank);

}

}

**Загрузка данных в объект из XML документа:**

**package** ru.icmit.oodb.lab3;

**import** ru.icmit.oodb.lab3.model.Bank;

**import** ru.icmit.oodb.lab3.model.BankAccount;

**import** ru.icmit.oodb.lab3.model.Client;

**import** java.util.ArrayList;

**import** java.util.List;

*/\*\**

*\* Пример, демонстрирующий преобразование XML документа в объект*

*\*/*

**public class** Lab3MainLoad {

**public static void** main(String[] args) {

Bank bank = Lab3XMLService.*loadBankFromXML*();

System.***out***.println(bank);

}

}

# Java Architecture for XML Binding (JAXB),

https://www.oracle.com/technical-resources/articles/javase/java-architecture-XML-binding.html