**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №7

по дисциплине: **«Визуальные средства разработки программных приложений»**

на тему: **«Обработка данных с использованием XML»**

Выполнил: студент группы ИТП-31

Пронуза М.Ю.

Проверил преподаватель:

Ястребов А.А.

Гомель 2024

**Цель работы**: изучить методы обработки данных с использованием XML в Java.

**Задание:**

Выполнить задания, сохраняя объекты приложения в одном или нескольких файлах XML. Четные варианты используют SAX-парсер, нечетные DOM-парсер

**Ход выполнения и результаты работы:**

Текстовый вариант задания для варианта 21 представлен на рисунке 1.

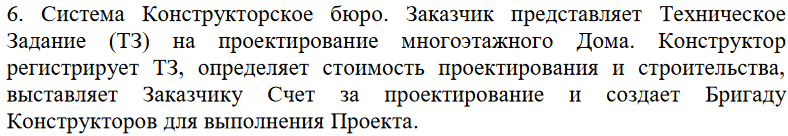


Рисунок 1 – Задание варианта 21

Для выполнения задания использована сериализация между классами. Была получена консольная программа.

На рисунке 2 представлено меню программы.

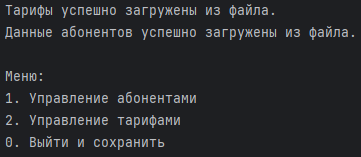


Рисунок 2 – Меню программы

На рисунке 3 представлен результат добавления тарифа.

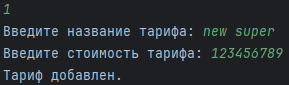


Рисунок 3 – Результат добавления тарифа

На рисунке 4 представлен вывод списка тарифов.



Рисунок 4 – Вывод списка тарифов

На рисунке 5 представлен вывод списка абонентов.

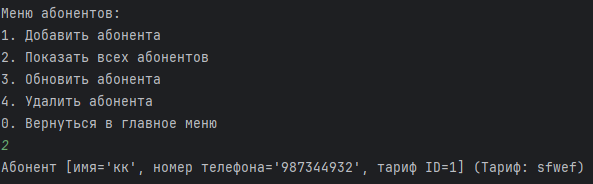


Рисунок 5 – Вывод списка абонентов

На рисунке 6 представлено редактирование абонентов.

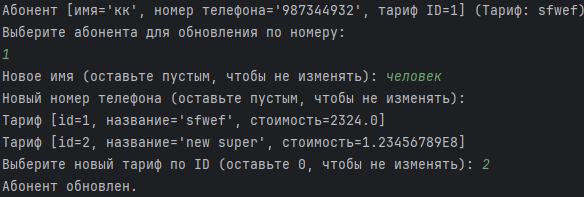


Рисунок 6 – редактирование абонентов

Текст *Main* представлен в приложении А.

Текст *XMLHandler* представлен в приложении Б.

Текст *Subscriber* представлен в приложении В.

Текст *Tariff* представлен в приложении Г.

Текст *SubscriberManager* представлен в приложении Д.

Текст *TariffManager* представлен в приложении Е.

**Вывод:** в ходе выполнения лабораторной работы были изучены методы обработки данных с использованием XML в Java.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Текст *Main***

import java.util.Scanner;

public class Main {

private static SubscriberManager subscriberManager;

private static TariffManager tariffManager;

private static final String SUBSCRIBERS\_FILE = "subscribers.xml";

private static final String TARIFFS\_FILE = "tariffs.xml";

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

tariffManager = new TariffManager();

subscriberManager = new SubscriberManager(tariffManager);

try {

// Попробуем загрузить абонентов из файла

subscriberManager.getSubscribers().addAll(XMLHandler.loadFromXML(SUBSCRIBERS\_FILE));

System.out.println("Данные абонентов успешно загружены из файла.");

} catch (Exception e) {

System.out.println("Ошибка при загрузке абонентов: " + e.getMessage());

}

int choice;

do {

System.out.println("\nМеню:");

System.out.println("1. Управление абонентами");

System.out.println("2. Управление тарифами");

System.out.println("0. Выйти и сохранить");

choice = scanner.nextInt();

scanner.nextLine(); // Очистка буфера после ввода числа

switch (choice) {

case 1:

manageSubscribers(scanner);

break;

case 2:

manageTariffs(scanner);

break;

case 0:

try {

XMLHandler.saveToXML(subscriberManager.getSubscribers(), SUBSCRIBERS\_FILE);

tariffManager.saveTariffs(); // Сохранить тарифы при выходе

System.out.println("Данные успешно сохранены.");

} catch (Exception e) {

System.out.println("Ошибка при сохранении данных: " + e.getMessage());

}

break;

default:

System.out.println("Неверный выбор, попробуйте снова.");

}

} while (choice != 0);

}

// Меню управления абонентами

private static void manageSubscribers(Scanner scanner) {

int choice;

do {

System.out.println("\nМеню абонентов:");

System.out.println("1. Добавить абонента");

System.out.println("2. Показать всех абонентов");

System.out.println("3. Обновить абонента");

System.out.println("4. Удалить абонента");

System.out.println("0. Вернуться в главное меню");

choice = scanner.nextInt();

scanner.nextLine();

switch (choice) {

case 1:

subscriberManager.addSubscriber(scanner);

break;

case 2:

subscriberManager.showAllSubscribers();

break;

case 3:

subscriberManager.updateSubscriber(scanner);

break;

case 4:

subscriberManager.deleteSubscriber(scanner);

break;

case 0:

break;

default:

System.out.println("Неверный выбор, попробуйте снова.");

}

} while (choice != 0);

}

// Меню управления тарифами

private static void manageTariffs(Scanner scanner) {

int choice;

do {

System.out.println("\nМеню тарифов:");

System.out.println("1. Добавить тариф");

System.out.println("2. Показать все тарифы");

System.out.println("3. Обновить тариф");

System.out.println("4. Удалить тариф");

System.out.println("0. Вернуться в главное меню");

choice = scanner.nextInt();

scanner.nextLine();

switch (choice) {

case 1:

tariffManager.addTariff(scanner);

break;

case 2:

tariffManager.showAllTariffs();

break;

case 3:

tariffManager.updateTariff(scanner);

break;

case 4:

tariffManager.deleteTariff(scanner);

break;

case 0:

break;

default:

System.out.println("Неверный выбор, попробуйте снова.");

}

} while (choice != 0);

}

}

**ПРИЛОЖЕНИЕ Б**

(обязательное)

**Текст *XMLHandler***

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.transform.OutputKeys;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerException;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.\*;

import java.io.File;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

public class XMLHandler {

// Загрузить абонентов из XML

public static List<Subscriber> loadFromXML(String filePath) throws Exception {

List<Subscriber> subscribers = new ArrayList<>();

File xmlFile = new File(filePath);

if (!xmlFile.exists()) {

throw new Exception("Файл не найден.");

}

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(xmlFile);

doc.getDocumentElement().normalize();

NodeList nodeList = doc.getElementsByTagName("subscriber");

for (int i = 0; i < nodeList.getLength(); i++) {

Element element = (Element) nodeList.item(i);

String name = element.getElementsByTagName("name").item(0).getTextContent();

String phoneNumber = element.getElementsByTagName("phoneNumber").item(0).getTextContent();

int tariffId = Integer.parseInt(element.getElementsByTagName("tariffId").item(0).getTextContent());

subscribers.add(new Subscriber(name, phoneNumber, tariffId));

}

return subscribers;

}

// Сохранить абонентов в XML

public static void saveToXML(List<Subscriber> subscribers, String filePath) throws ParserConfigurationException, TransformerException {

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.newDocument();

Element rootElement = doc.createElement("subscribers");

doc.appendChild(rootElement);

for (Subscriber subscriber : subscribers) {

Element subscriberElement = doc.createElement("subscriber");

rootElement.appendChild(subscriberElement);

Element nameElement = doc.createElement("name");

nameElement.appendChild(doc.createTextNode(subscriber.getName()));

subscriberElement.appendChild(nameElement);

Element phoneElement = doc.createElement("phoneNumber");

phoneElement.appendChild(doc.createTextNode(subscriber.getPhoneNumber()));

subscriberElement.appendChild(phoneElement);

Element tariffIdElement = doc.createElement("tariffId");

tariffIdElement.appendChild(doc.createTextNode(String.valueOf(subscriber.getTariffId())));

subscriberElement.appendChild(tariffIdElement);

}

TransformerFactory transformerFactory = TransformerFactory.newInstance();

Transformer transformer = transformerFactory.newTransformer();

transformer.setOutputProperty(OutputKeys.INDENT, "yes");

DOMSource source = new DOMSource(doc);

StreamResult result = new StreamResult(new File(filePath));

transformer.transform(source, result);

}

// Загрузить тарифы из XML

public static List<Tariff> loadTariffsFromXML(String filePath) throws Exception {

List<Tariff> tariffs = new ArrayList<>();

File xmlFile = new File(filePath);

if (!xmlFile.exists()) {

throw new Exception("Файл не найден.");

}

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(xmlFile);

doc.getDocumentElement().normalize();

NodeList nodeList = doc.getElementsByTagName("tariff");

for (int i = 0; i < nodeList.getLength(); i++) {

Element element = (Element) nodeList.item(i);

String name = element.getElementsByTagName("name").item(0).getTextContent();

double cost = Double.parseDouble(element.getElementsByTagName("cost").item(0).getTextContent());

tariffs.add(new Tariff(name, cost));

}

return tariffs;

}

// Сохранить тарифы в XML

public static void saveTariffsToXML(List<Tariff> tariffs, String filePath) throws ParserConfigurationException, TransformerException {

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.newDocument();

Element rootElement = doc.createElement("tariffs");

doc.appendChild(rootElement);

for (Tariff tariff : tariffs) {

Element tariffElement = doc.createElement("tariff");

rootElement.appendChild(tariffElement);

Element nameElement = doc.createElement("name");

nameElement.appendChild(doc.createTextNode(tariff.getName()));

tariffElement.appendChild(nameElement);

Element costElement = doc.createElement("cost");

costElement.appendChild(doc.createTextNode(String.valueOf(tariff.getCost())));

tariffElement.appendChild(costElement);

}

TransformerFactory transformerFactory = TransformerFactory.newInstance();

Transformer transformer = transformerFactory.newTransformer();

transformer.setOutputProperty(OutputKeys.INDENT, "yes");

DOMSource source = new DOMSource(doc);

StreamResult result = new StreamResult(new File(filePath));

transformer.transform(source, result);

}

}

**ПРИЛОЖЕНИЕ В**

(обязательное)

**Текст *Subscriber***

public class Subscriber {

private String name;

private String phoneNumber;

private int tariffId; // Ссылка на тариф по ID

public Subscriber(String name, String phoneNumber, int tariffId) {

this.name = name;

this.phoneNumber = phoneNumber;

this.tariffId = tariffId;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPhoneNumber() {

return phoneNumber;

}

public void setPhoneNumber(String phoneNumber) {

this.phoneNumber = phoneNumber;

}

public int getTariffId() {

return tariffId;

}

public void setTariffId(int tariffId) {

this.tariffId = tariffId;

}

@Override

public String toString() {

return "Абонент [имя='" + name + "', номер телефона='" + phoneNumber + "', тариф ID=" + tariffId + "]";

}

}

**ПРИЛОЖЕНИЕ Г**

(обязательное)

**Текст *Tariff***

public class Tariff {

private static int idCounter = 0; // Счетчик для создания уникальных идентификаторов

private int id; // Уникальный идентификатор тарифа

private String name;

private double cost;

public Tariff(String name, double cost) {

this.id = ++idCounter;

this.name = name;

this.cost = cost;

}

public int getId() {

return id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getCost() {

return cost;

}

public void setCost(double cost) {

this.cost = cost;

}

@Override

public String toString() {

return "Тариф [id=" + id + ", название='" + name + "', стоимость=" + cost + "]";

}

}

**ПРИЛОЖЕНИЕ Д**

(обязательное)

**Текст *SubscriberManager***

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class SubscriberManager {

private List<Subscriber> subscribers;

private TariffManager tariffManager;

public SubscriberManager(TariffManager tariffManager) {

this.subscribers = new ArrayList<>();

this.tariffManager = tariffManager;

}

// Добавить абонента

public void addSubscriber(Scanner scanner) {

System.out.print("Введите имя абонента: ");

String name = scanner.nextLine();

System.out.print("Введите номер телефона: ");

String phoneNumber = scanner.nextLine();

tariffManager.showAllTariffs();

System.out.print("Выберите тариф по ID: ");

int tariffId = scanner.nextInt();

scanner.nextLine();

if (tariffManager.getTariffById(tariffId) != null) {

subscribers.add(new Subscriber(name, phoneNumber, tariffId));

System.out.println("Абонент добавлен.");

} else {

System.out.println("Неверный выбор тарифа.");

}

}

// Показать всех абонентов

public void showAllSubscribers() {

if (subscribers.isEmpty()) {

System.out.println("Список абонентов пуст.");

} else {

for (Subscriber subscriber : subscribers) {

Tariff tariff = tariffManager.getTariffById(subscriber.getTariffId());

System.out.println(subscriber + " (Тариф: " + (tariff != null ? tariff.getName() : "не найден") + ")");

}

}

}

// Обновить абонента

public void updateSubscriber(Scanner scanner) {

showAllSubscribers();

System.out.print("Выберите абонента для обновления по номеру: ");

int subscriberIndex = scanner.nextInt();

scanner.nextLine();

if (subscriberIndex >= 1 && subscriberIndex <= subscribers.size()) {

Subscriber subscriber = subscribers.get(subscriberIndex - 1);

System.out.print("Новое имя (оставьте пустым, чтобы не изменять): ");

String newName = scanner.nextLine();

if (!newName.isEmpty()) {

subscriber.setName(newName);

}

System.out.print("Новый номер телефона (оставьте пустым, чтобы не изменять): ");

String newPhoneNumber = scanner.nextLine();

if (!newPhoneNumber.isEmpty()) {

subscriber.setPhoneNumber(newPhoneNumber);

}

tariffManager.showAllTariffs();

System.out.print("Выберите новый тариф по ID (оставьте 0, чтобы не изменять): ");

int tariffId = scanner.nextInt();

scanner.nextLine();

if (tariffId > 0 && tariffManager.getTariffById(tariffId) != null) {

subscriber.setTariffId(tariffId);

}

System.out.println("Абонент обновлен.");

} else {

System.out.println("Неверный выбор.");

}

}

// Удалить абонента

public void deleteSubscriber(Scanner scanner) {

showAllSubscribers();

System.out.print("Выберите абонента для удаления по номеру: ");

int subscriberIndex = scanner.nextInt();

scanner.nextLine();

if (subscriberIndex >= 1 && subscriberIndex <= subscribers.size()) {

subscribers.remove(subscriberIndex - 1);

System.out.println("Абонент удален.");

} else {

System.out.println("Неверный выбор.");

}

}

public List<Subscriber> getSubscribers() {

return subscribers;

}

}

**ПРИЛОЖЕНИЕ Е**

(обязательное)

**Текст *TariffManager***

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class TariffManager {

private List<Tariff> tariffs;

private static final String XML\_FILE = "tariffs.xml";

public TariffManager() {

this.tariffs = new ArrayList<>();

loadTariffs();

}

// Загрузить тарифы из XML

public void loadTariffs() {

try {

tariffs = XMLHandler.loadTariffsFromXML(XML\_FILE);

System.out.println("Тарифы успешно загружены из файла.");

} catch (Exception e) {

System.out.println("Ошибка при загрузке тарифов: " + e.getMessage());

}

}

// Сохранить тарифы в XML

public void saveTariffs() {

try {

XMLHandler.saveTariffsToXML(tariffs, XML\_FILE);

System.out.println("Тарифы успешно сохранены в файл.");

} catch (Exception e) {

System.out.println("Ошибка при сохранении тарифов: " + e.getMessage());

}

}

// Добавить тариф

public void addTariff(Scanner scanner) {

System.out.print("Введите название тарифа: ");

String name = scanner.nextLine();

System.out.print("Введите стоимость тарифа: ");

double cost = scanner.nextDouble();

scanner.nextLine();

tariffs.add(new Tariff(name, cost));

System.out.println("Тариф добавлен.");

}

// Показать все тарифы

public void showAllTariffs() {

if (tariffs.isEmpty()) {

System.out.println("Список тарифов пуст.");

} else {

for (Tariff tariff : tariffs) {

System.out.println(tariff);

}

}

}

// Обновить тариф

public void updateTariff(Scanner scanner) {

showAllTariffs();

System.out.print("Выберите тариф для обновления по ID: ");

int tariffId = scanner.nextInt();

scanner.nextLine();

Tariff tariff = getTariffById(tariffId);

if (tariff != null) {

System.out.print("Новое название (оставьте пустым, чтобы не изменять): ");

String newName = scanner.nextLine();

if (!newName.isEmpty()) {

tariff.setName(newName);

}

System.out.print("Новая стоимость (оставьте пустым, чтобы не изменять): ");

String newCost = scanner.nextLine();

if (!newCost.isEmpty()) {

tariff.setCost(Double.parseDouble(newCost));

}

System.out.println("Тариф обновлен.");

} else {

System.out.println("Неверный выбор.");

}

}

// Удалить тариф

public void deleteTariff(Scanner scanner) {

showAllTariffs();

System.out.print("Выберите тариф для удаления по ID: ");

int tariffId = scanner.nextInt();

scanner.nextLine();

Tariff tariff = getTariffById(tariffId);

if (tariff != null) {

tariffs.remove(tariff);

System.out.println("Тариф удален.");

} else {

System.out.println("Неверный выбор.");

}

}

// Получить тариф по ID

public Tariff getTariffById(int id) {

for (Tariff tariff : tariffs) {

if (tariff.getId() == id) {

return tariff;

}

}

return null; // Тариф не найден

}

public List<Tariff> getTariffs() {

return tariffs;

}

}