**ЛАБОРАТОРНАЯ РАБОТА №9**

**XML: ОСНОВЫ СОЗДАНИЯ И ОБРАБОТКИ СТРУКТУРИРОВАННЫХ ДАННЫХ**

Цель работы: освоить принципы создания, валидации и обработки XML документов, развить навыки анализа данных и работы с инструментами для обработки структурированных форматов.

Необходимая теоретическая подготовка: основы XML (структура, теги, атрибуты). Понятие валидации и схем (XSD). Представление о парсинге XML с использованием языков программирования.

**1. Создание XML-документа:**

Был создан XML-документ *library.xml,* представляющий библиотеку, состоящую из трёх книг. Каждая книга содержит элементы: *title, author, year, genre, price* и уникальный атрибут *id.* Документ соответствует синтаксису XML.

**2. Создание XSD-схемы:**

Разработана XSD-схема *library.xsd* для проверки структуры и данных *library.xml.* Схема проверяет обязательное наличие всех элементов, уникальность и обязательность атрибута *id,* а также ограничения на диапазон года (1800–2025) и формат цены (положительное число с двумя знаками после запятой).

**3. Валидация XML-документа:**

Для обеспечения соответствия XML-документа (library.xml) строго определённым правилам, заданным в XSD-схеме (library.xsd), была написана Java-программа. Цель валидации — гарантировать, что структура XML и значения его элементов соответствуют требованиям, описанным в схеме.

В основе программы лежит стандартная библиотека Java для валидации XML (javax.xml.validation). Это обеспечивает надёжность и совместимость с XML Schema Definition (XSD).

Для реализации применяется класс SchemaFactory, который создаёт схему на основе XSD. Затем используется объект Validator для проверки XML-документа.

Сначала загружаются файлы library.xml и library.xsd с помощью File. На основе XSD создаётся схема (Schema) через SchemaFactory. Объект Validator, полученный из схемы, используется для проверки XML на соответствие требованиям.

Если в документе находятся ошибки (например, отсутствует обязательный элемент или значение не соответствует типу), программа выводит описание ошибки. Если ошибок нет, выводится сообщение о валидности документа.

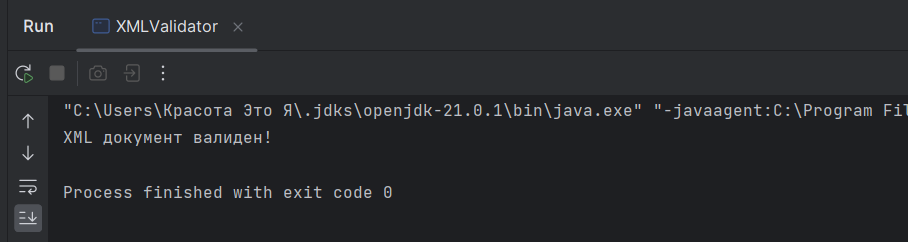


Рисунок 1.1 – Результат работы валидации

**4. Обработка XML-документа:**

Создана Java-программа для обработки *library.xml.*

Программа:

* Читает данные из XML-документа.
* Выводит список всех книг и их характеристики в консоль.
* Вычисляет среднюю цену книг.
* Фильтрует книги по жанру *Программирование* и выводит результат.

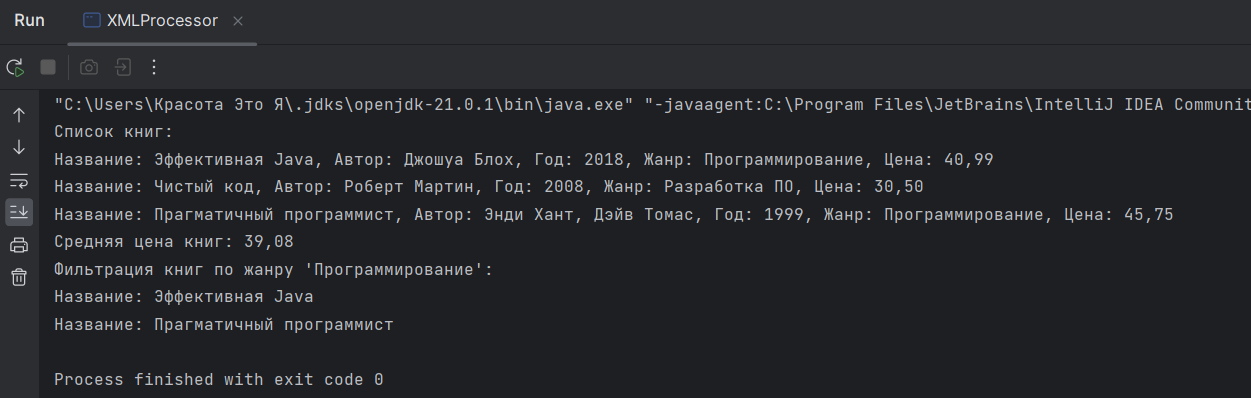


Рисунок 1.2 – **Результат обработки XML-документа**

**5. Исследование и сравнение:**

1. **Корневой элемент** <rss>**:** является корневым элементом документа и содержит атрибут version="2.0", указывающий версию стандарта RSS. Этот элемент служит контейнером для всей информации RSS-ленты.
2. **Элемент** <channel>**:** подчинён корневому элементу <rss> и содержит описание канала. Это основной контейнер, в котором представлены основные свойства канала и список новостей.
3. **Внутри** <channel>**:**

* <title>: Отображает название канала (в данном случае "Новости технологий").
* <link>: Указывает ссылку на главную страницу новостного источника.
* <description>: Даёт краткое описание канала (например, "Последние новости из мира технологий").

1. **Элемент** <item>**:** каждый <item> внутри <channel> представляет отдельную новость. Это повторяющийся элемент, который может встречаться несколько раз (в данном примере два <item>).
2. **Внутри** <item>**:**

* <title>: Название новости, например, "Новый смартфон от компании X".
* <link>: Ссылка на страницу с полным текстом новости.
* <description>: Краткое описание новости.
* <pubDate>: Дата публикации новости в формате RFC 822 (например, "Mon, 21 Apr 2025 09:00:00 GMT").

1. **Вложенность:**

<rss> → <channel> → <item> → элементы <title>, <link>, <description>, <pubDate>. Такая структура позволяет логически упорядочивать данные и легко их извлекать.

1. **Атрибуты:** атрибуты в этом документе минимальны, что характерно для RSS. Пример: version="2.0" у <rss>.

Этот XML-документ представляет собой RSS-ленту, которая используется для автоматического предоставления обновлений контента пользователям. Программы-агрегаторы или RSS-ридеры используют такие документы, чтобы отображать последние новости или статьи без необходимости вручную заходить на веб-сайт. Например, данный документ можно использовать для информирования о новых статьях на тему технологий.