**ЛАБОРАТОРНАЯ РАБОТА № 10**

«Введение в работу с XML, JSON, HTML и Excel»

**Цель:** получить представление о механизме парсинга в языке программирования Java

УЧЕБНЫЕ ВОПРОСЫ

[1 ПРИМЕР РЕШЕНИЯ ЗАДАЧ 2](#_Toc129207843)

[1.1 Краткая теория 2](#_Toc129207844)

[1.2 Пример работы с XML 3](#_Toc129207845)

[1.2.1 Как создать файл XML 3](#_Toc129207846)

[1.2.2 Как прочитать файл XML 4](#_Toc129207847)

[1.2.3 Пример файла XML 5](#_Toc129207848)

[1.2.4 Ссылки на документацию 5](#_Toc129207849)

[1.3 Пример работы с JSON 6](#_Toc129207850)

[1.3.1 Как подключить библиотеку в виде jar файла? 6](#_Toc129207851)

[1.3.2 Как создать файл JSON 9](#_Toc129207852)

[1.3.3 Как прочитать файл JSON 10](#_Toc129207853)

[1.3.4 Пример файла json 10](#_Toc129207854)

[1.3.5 Ссылки на документацию 11](#_Toc129207855)

[1.4 Пример работы с HTML 12](#_Toc129207856)

[1.4.1 Как подключить библиотеку c помощью maven? 12](#_Toc129207857)

[1.4.2 Пример парсинга страницы https://itlearn.ru/first-steps 14](#_Toc129207858)

[1.4.3 Пример парсинга страницы http://fat.urfu.ru/index.html 15](#_Toc129207859)

[1.4.4 Ссылки на документацию 16](#_Toc129207860)

[1.5 Пример работы с Excel 17](#_Toc129207861)

[1.5.1 Подключение библиотек c помощью maven 17](#_Toc129207862)

[1.5.2 Как создать файл Excel 18](#_Toc129207863)

[1.5.3 Как прочитать файл Excel 20](#_Toc129207864)

[1.5.4 Пример таблицы Excel 21](#_Toc129207865)

[1.5.5 Ссылки на документацию 21](#_Toc129207866)

[2 ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ 22](#_Toc129207867)

[2.1 Выполните все примеры из раздела 1 22](#_Toc129207868)

[2. 2 Выполните следующие доработки проекта XML парсера: 22](#_Toc129207869)

[Подсказки: 22](#_Toc129207870)

[2.2 Выполните следующие доработки проекта JSON парсера: 23](#_Toc129207871)

[Подсказки: 23](#_Toc129207872)

[2.3 Выполните следующие доработки проекта HTML парсера 24](#_Toc129207873)

[2.4 Выполните следующую доработки проекта Excel парсера 25](#_Toc129207874)

[3 ОПИСАНИЕ РЕЗУЛЬТАТА ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ 26](#_Toc129207875)

# 1 ПРИМЕР РЕШЕНИЯ ЗАДАЧ

## 1.1 Краткая теория

Парсинг XML, JSON и Excel и HTML являются распространенными операциями в программировании. Они позволяют получать данные из файлов или сетевых источников и обрабатывать их в программе.

Парсинг XML - это процесс извлечения данных из XML-документа, который является стандартным форматом обмена данными между различными системами. Для парсинга XML в Java можно использовать библиотеки, такие как JAXP, DOM, SAX или StAX.

Парсинг JSON - это процесс извлечения данных из JSON-документа, который является более легковесным и удобочитаемым форматом, чем XML. JSON-документы могут содержать массивы, объекты и примитивные типы данных. Для парсинга JSON в Java можно использовать библиотеки, такие как Gson, Jackson или org.json.

Парсинг Excel - это процесс извлечения данных из таблицы Excel, которая может содержать данные в различных форматах, таких как текст, числа, даты и формулы. Для парсинга Excel в Java можно использовать библиотеки, такие как Apache POI или JExcelAPI.

JSON (JavaScript Object Notation) — это формат обмена данными, основанный на языке JavaScript, который широко используется в веб-разработке. JSON-документы легко читаются и создаются человеком и могут быть использованы для передачи данных между клиентом и сервером. Они содержат пары "ключ-значение" и могут быть представлены в виде объектов, массивов или примитивных типов данных.

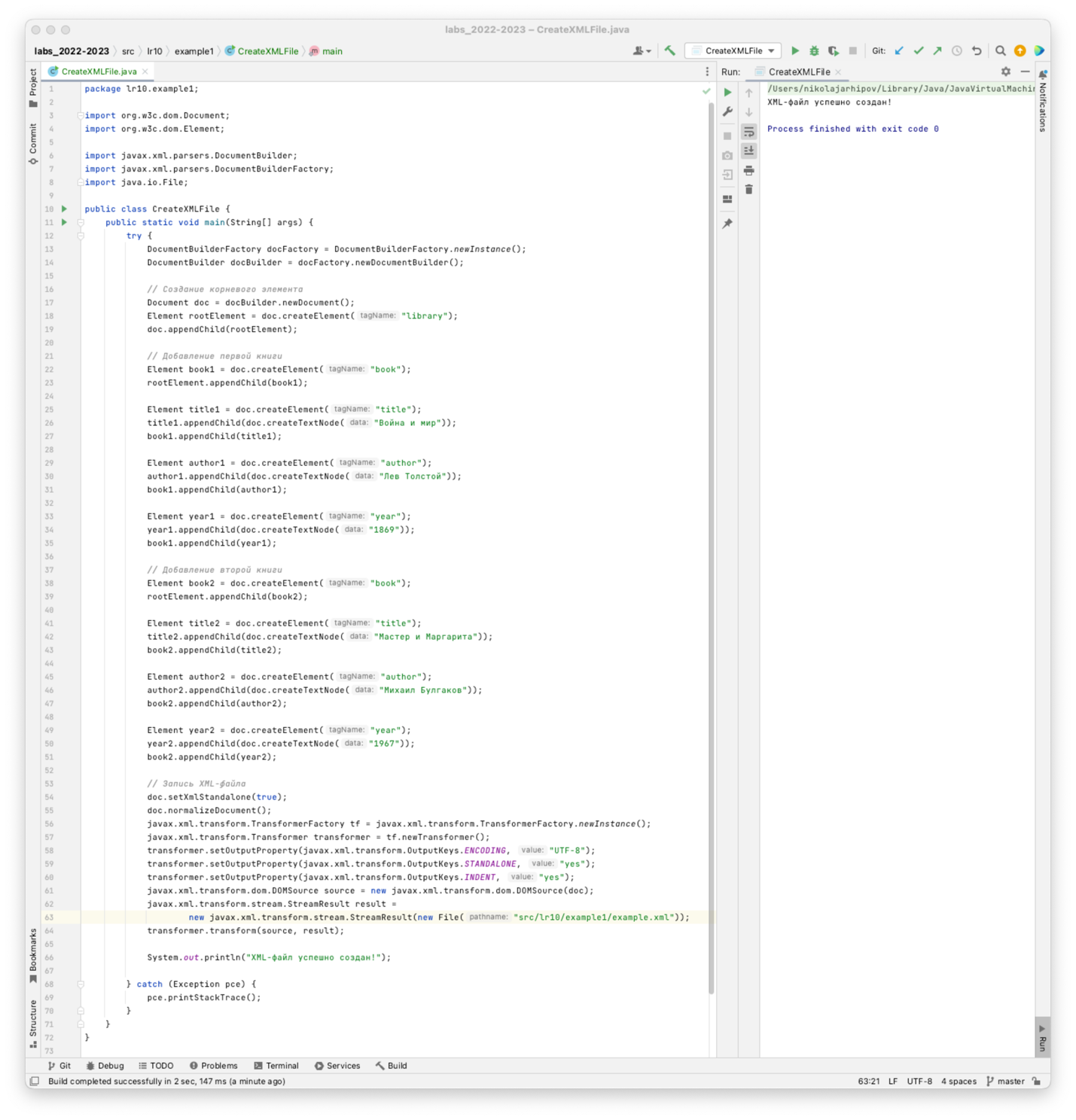
Для парсинга XML, JSON и Excel в Java 8 и выше можно использовать следующие библиотеки:

* JAXP (Java API for XML Processing) - для парсинга XML
* Gson - для парсинга и создания JSON
* Jackson - для парсинга и создания JSON
* org.json - для парсинга и создания JSON
* Apache POI - для парсинга Excel

Использование этих библиотек может значительно упростить процесс обработки данных в Java и повысить эффективность работы с файлами и сетевыми источниками данных.

## 1.2 Пример работы с XML

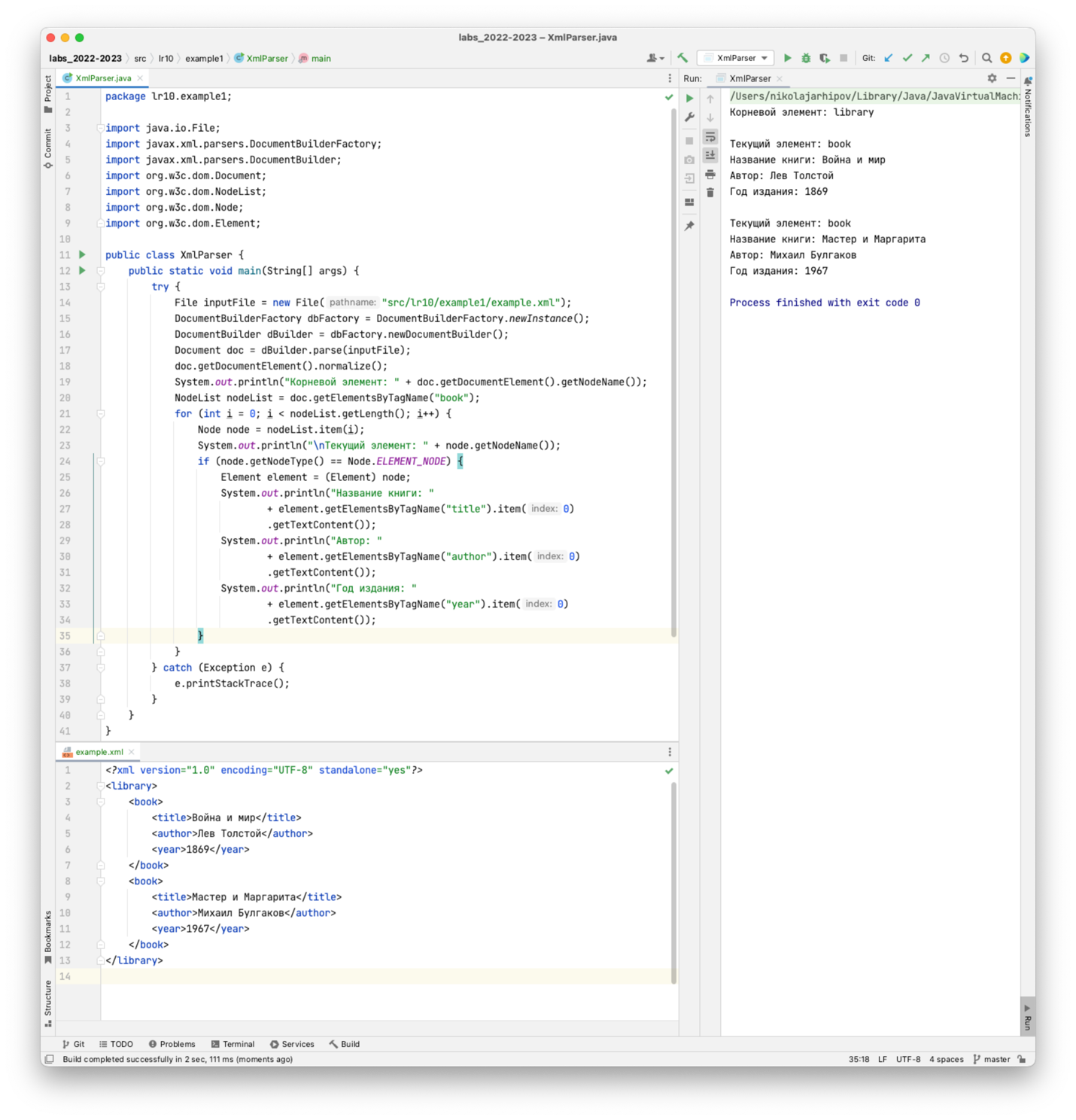
1.2.1 Как создать файл XML



Сначала мы создаем экземпляр DocumentBuilderFactory и DocumentBuilder, чтобы получить объект Document, который является корневым элементом нашего XML-документа. Затем мы создаем элементы и добавляем их в документ, вызывая методы createElement и appendChild на соответствующих элементах. После того, как мы создали и заполнили XML-документ, мы используем Transformer для записи DOM-структуры в файл.

Как только мы завершаем запись, мы получаем XML-файл с содержимым, которое мы определили в нашей программе.

### 1.2.2 Как прочитать файл XML



Эта программа на Java использует библиотеку **javax.xml.parsers** для чтения и анализа XML-файла. Программа имеет класс **XmlParser**, в котором есть единственный метод **main**, который является точкой входа в программу.

Когда программа запускается, она открывает XML-файл **src/lr10/example.xml**, используя класс **java.io.File**, который представляет файл на диске. Затем программа создает экземпляр класса **javax.xml.parsers.DocumentBuilderFactory**, который используется для создания **javax.xml.parsers.DocumentBuilder**, который, в свою очередь, используется для создания **org.w3c.dom.Document**.

Далее, программа вызывает метод **parse** у экземпляра **DocumentBuilder**, передавая в качестве аргумента **File**, который был создан ранее. Этот метод читает XML-файл и возвращает объект **Document**, который представляет собой древовидное представление содержимого XML-файла.

Затем программа вызывает метод **getDocumentElement** у объекта **Document**, чтобы получить корневой элемент XML-документа. С помощью метода **getNodeName** программа выводит имя корневого элемента на консоль.

Затем программа получает все элементы **book** из XML-документа, используя метод **getElementsByTagName** у объекта **Document**. Затем программа проходит циклом по этим элементам и выводит на консоль название книги, имя автора и год издания, используя методы **getElementsByTagName** и **getTextContent** у объектов **Element**.

### 1.2.3 Пример файла XML

*<?*xml version="1.0" encoding="UTF-8" standalone="yes"*?>*<library>  
 <book>  
 <title>Война и мир</title>  
 <author>Лев Толстой</author>  
 <year>1869</year>  
 </book>  
 <book>  
 <title>Мастер и Маргарита</title>  
 <author>Михаил Булгаков</author>  
 <year>1967</year>  
 </book>  
</library>

### 1.2.4 Ссылки на документацию

1. [Документация по библиотеке DOM](https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/package-summary.html) - этот ресурс описывает интерфейсы и классы, используемые для создания и манипулирования XML-документами с помощью DOM.
2. [Документация по классу DocumentBuilderFactory](https://docs.oracle.com/javase/8/docs/api/javax/xml/parsers/DocumentBuilderFactory.html) - этот ресурс описывает класс DocumentBuilderFactory, который я использовал для получения экземпляра DocumentBuilder.
3. [Документация по классу DocumentBuilder](https://docs.oracle.com/javase/8/docs/api/javax/xml/parsers/DocumentBuilder.html) - этот ресурс описывает класс DocumentBuilder, который я использовал для создания DOM-структуры.
4. [Документация по классу Element](https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/Element.html" \t "_new) - этот ресурс описывает класс Element, который я использовал для создания элементов XML-документа.
5. [Документация по классу Transformer](https://docs.oracle.com/javase/8/docs/api/javax/xml/transform/Transformer.html) - этот ресурс описывает класс Transformer, который я использовал для записи DOM-структуры в XML-файл.

## 1.3 Пример работы с JSON

### 1.3.1 Как подключить библиотеку в виде jar файла?

Для использования библиотеки org.json.simple в вашем проекте вам нужно добавить зависимость для этой библиотеки в файл pom.xml (если вы используете Maven) или добавить jar-файл этой библиотеки в свой проект (если вы не используете Maven). В данной работе не используется Maven напрямую.

Если вы используете Maven, добавьте следующую зависимость в ваш файл pom.xml:

<dependency>

<groupId>com.googlecode.json-simple</groupId>

<artifactId>json-simple</artifactId>

<version>1.1.1</version>

</dependency>

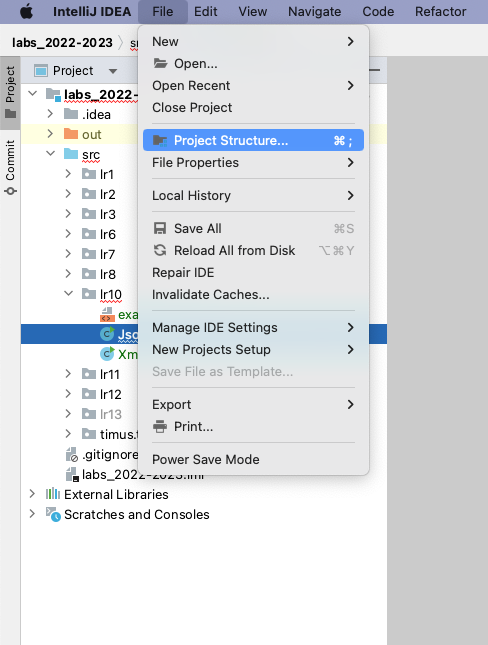
Если вы не используете Maven, загрузите jar-файл библиотеки с официального сайта и добавьте его в свой проект. Вы можете найти jar-файл библиотеки на странице загрузки: <https://code.google.com/archive/p/json-simple/downloads>.

После того, как вы добавили зависимость или jar-файл в свой проект, вы можете использовать **org.json.simple** в своем коде, например:

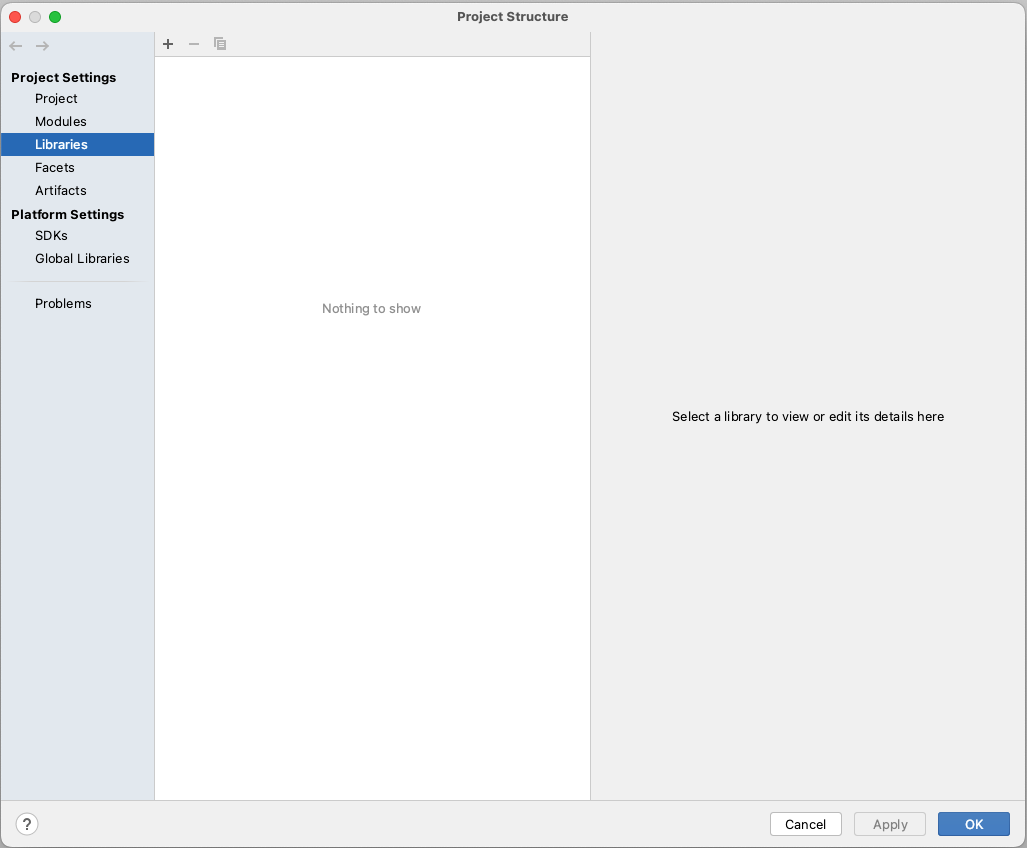
import org.json.simple.JSONArray;

import org.json.simple.JSONObject;

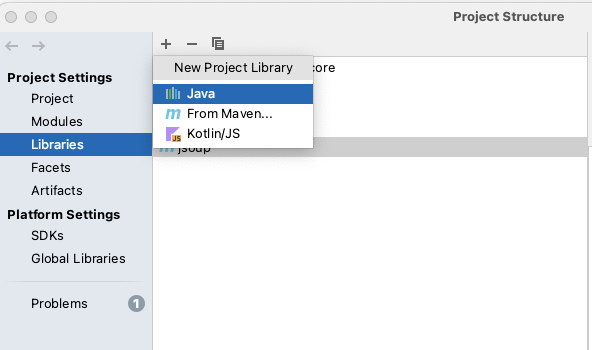
Для того чтобы добавть библиотеку в IntelliJ IDEA в виде jar файла необходмо выбрать File – Project Structure, как показано на рисунке ниже.



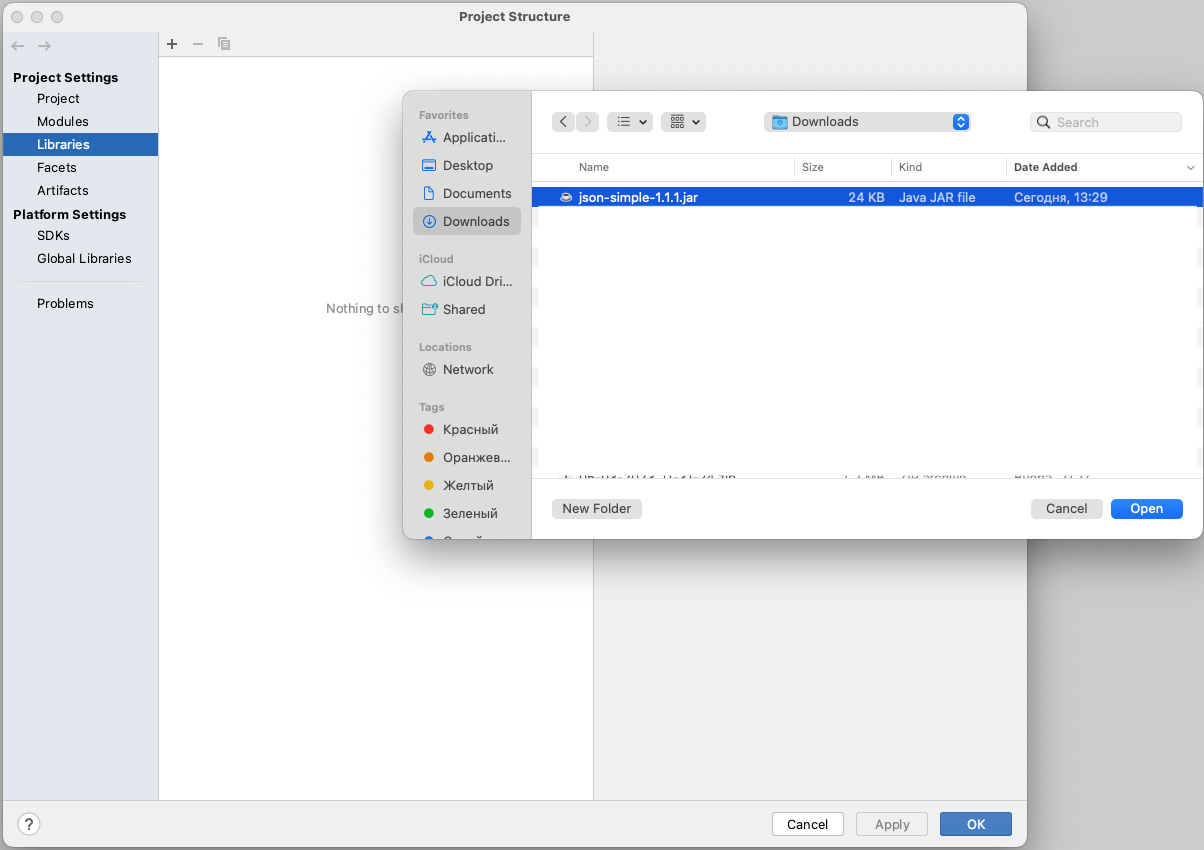
В окрывшемся окне нужно выбрать раздел Libraries, как показано на рисунке ниже.



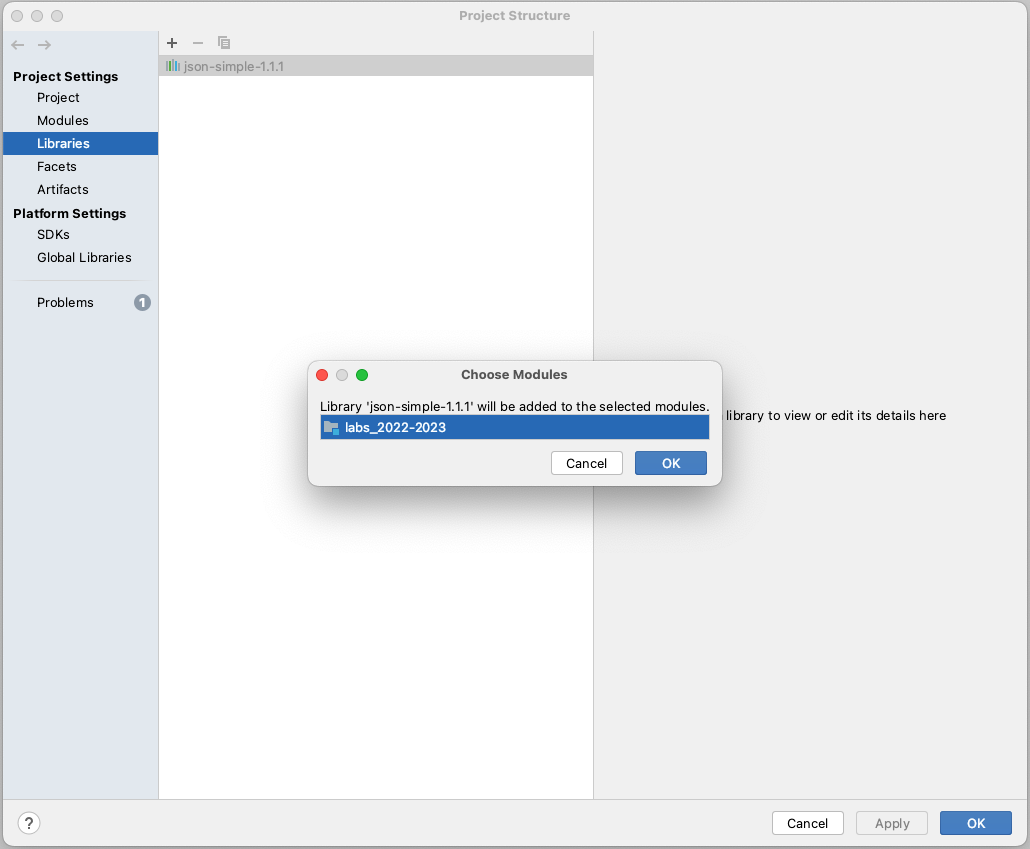
Нажать на кнопку «+» и выбрать Java, как показано на рисунке ниже.



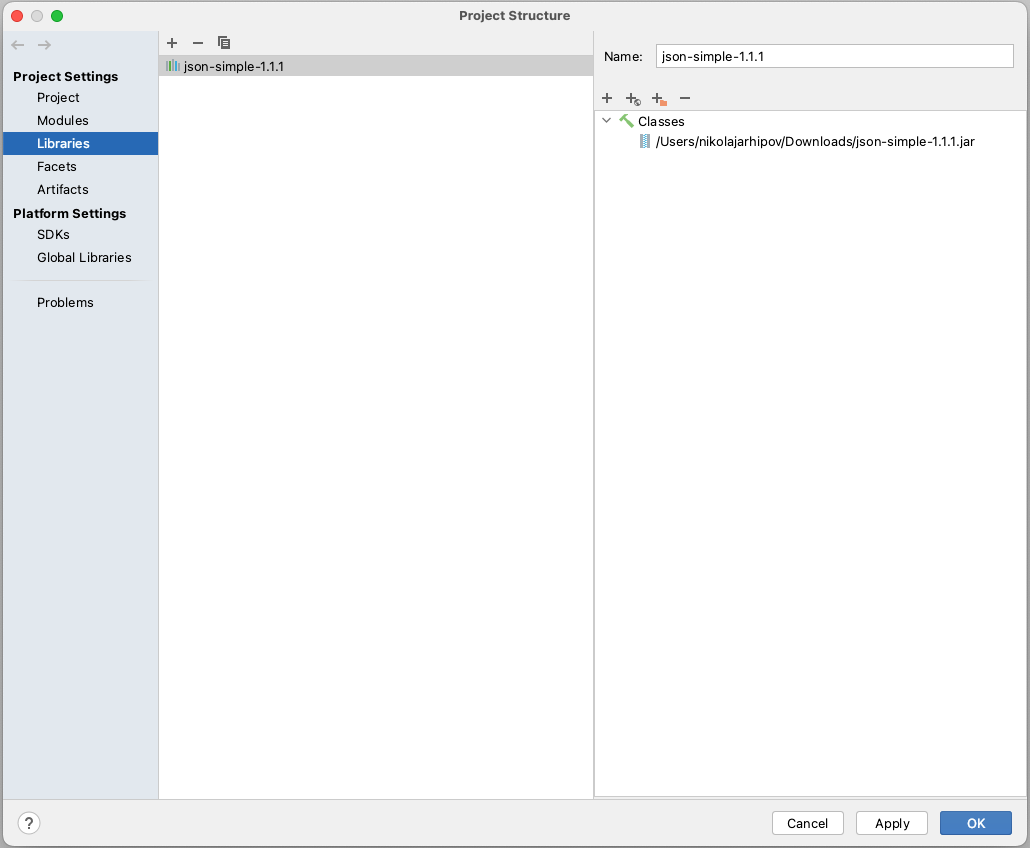
Затем нужно выбрать скачанный ранее файл в формате jar библиотеки json-simple, как показано на рисунке ниже.



Появиться окно «Choose Modules», нажмите «ОК», как показано на рисунке ниже.

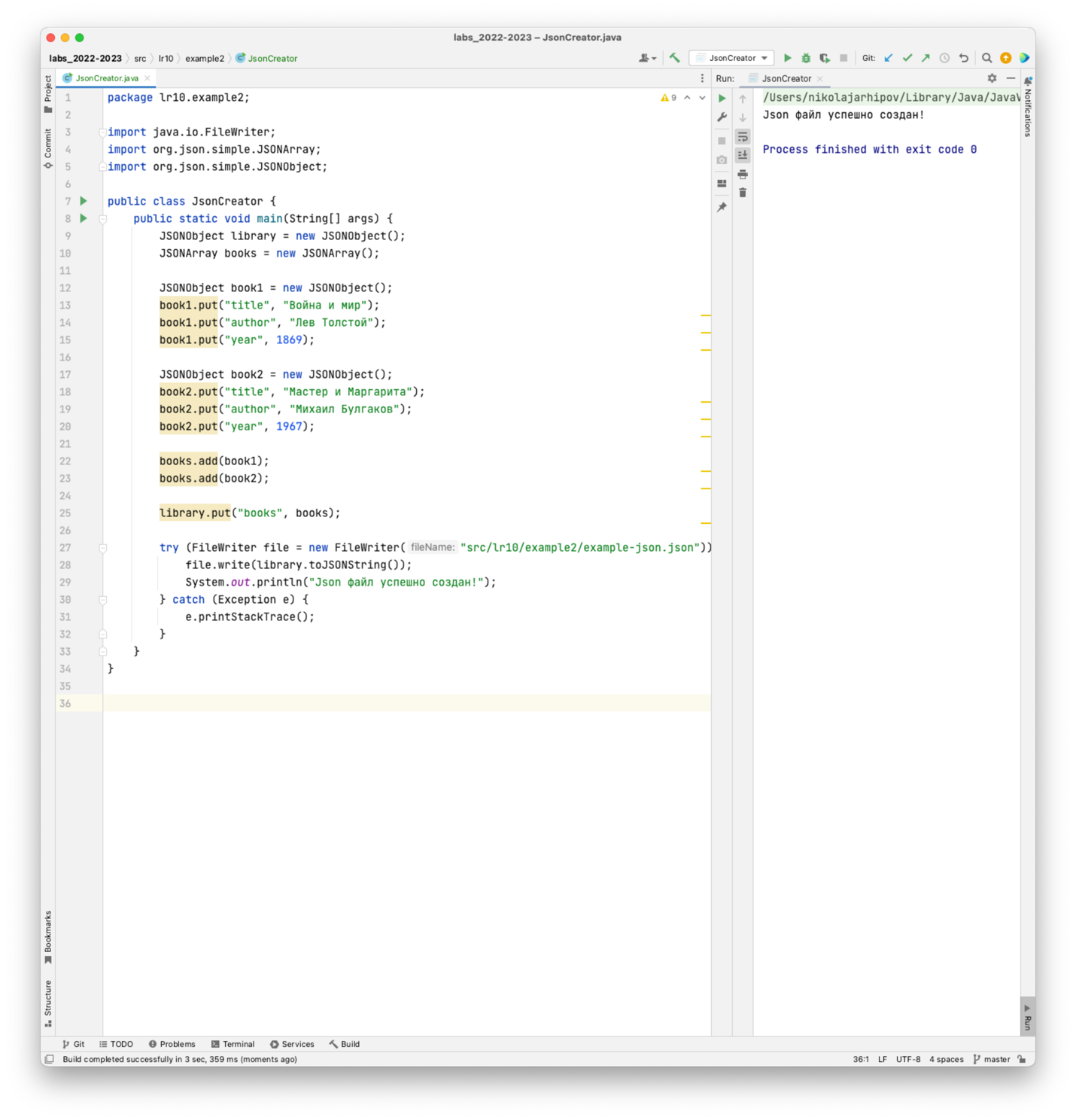


Если добавление прошло успешно, то результат будет как на рисунке ниже.



Поздравляю, библиотека jsoup добавлена успешно.

### 1.3.2 Как создать файл JSON

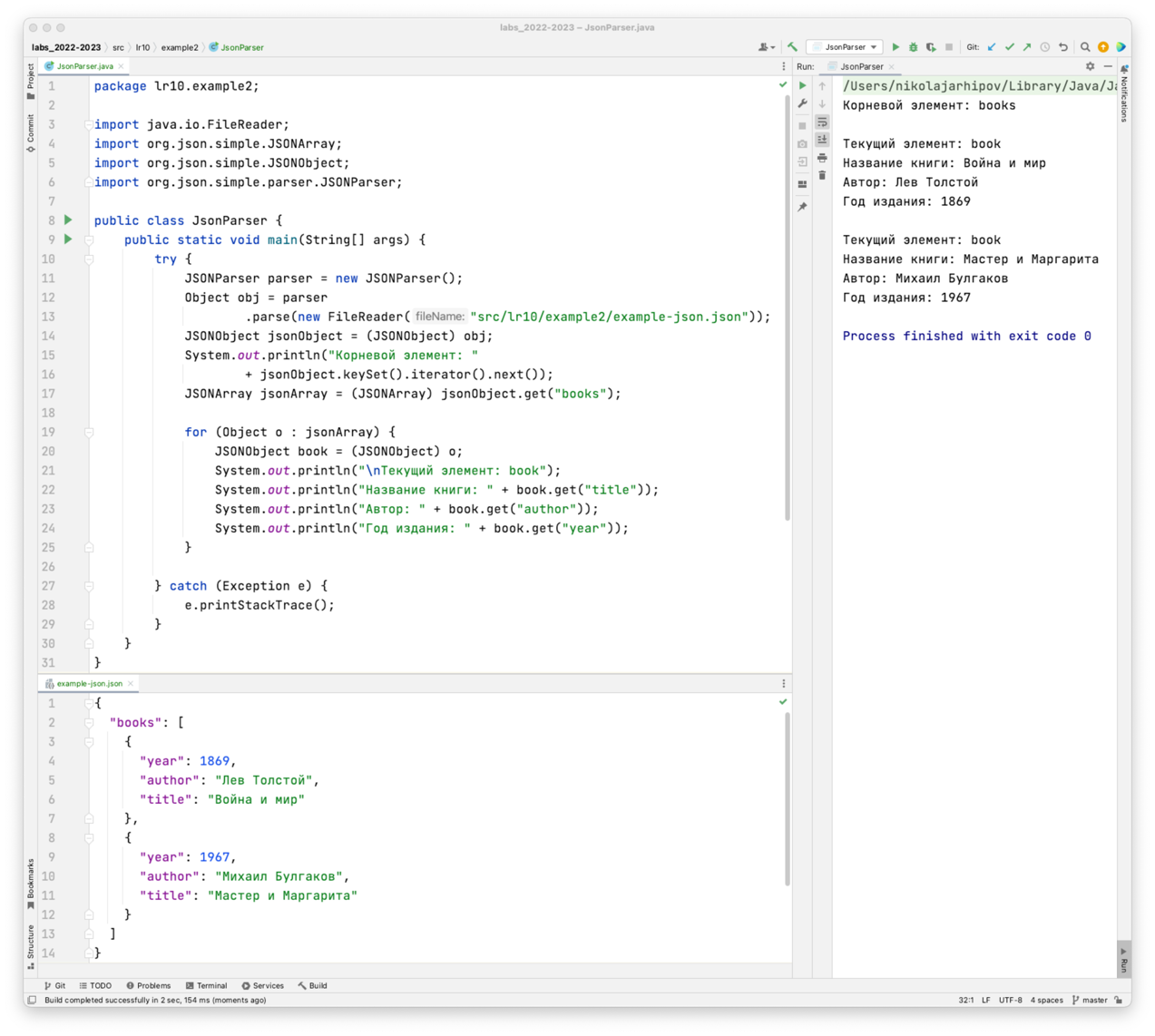


В этом примере мы создаем JSONObject, который представляет нашу библиотеку, и JSONArray, который содержит объекты JSONObject, представляющие каждую книгу. Мы добавляем две книги в наш массив, а затем добавляем этот массив в объект JSONObject, который представляет всю библиотеку.

После этого мы записываем наш JSON-объект в файл **"example-json.json"** с помощью FileWriter.

Если все идет успешно, мы получаем сообщение об успешном создании Json файла в консоли.

### 1.3.3 Как прочитать файл JSON



Здесь мы используем библиотеку JSON.simple для работы с файлом JSON. Мы создаем экземпляр класса JSONParser, который используется для чтения файла JSON. Затем мы преобразуем объект в JSONObject и получаем ключ корневого элемента. Мы получаем массив книг, проходим по каждой книге и получаем значения полей.

### 1.3.4 Пример файла json

{  
 "books": [  
 {  
 "title": "Java Programming",  
 "author": "John Doe",  
 "year": "2015"  
 },  
 {  
 "title": "Python Programming",  
 "author": "Jane Smith",  
 "year": "2018"  
 },  
 {  
 "title": "Ruby Programming",  
 "author": "Bob Johnson",  
 "year": "2016"  
 }  
 ]  
}

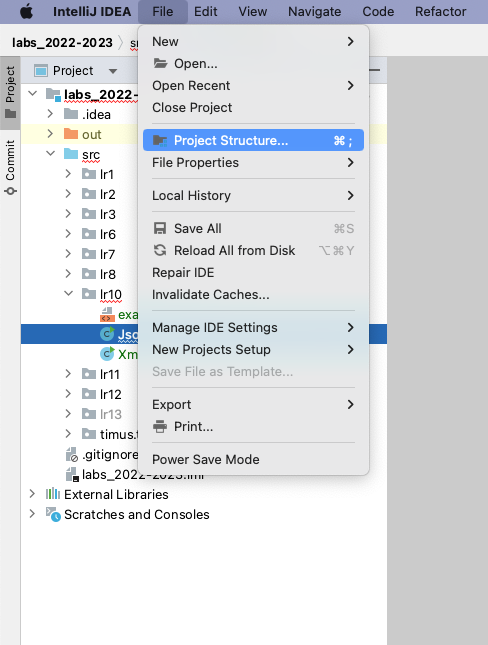
### 1.3.5 Ссылки на документацию

1. Официальный сайт библиотеки JSON.simple: <https://github.com/fangyidong/json-simple>
2. Документация JSONObject: <https://static.javadoc.io/com.googlecode.json-simple/json-simple/1.1.1/org/json/simple/JSONObject.html>
3. Документация JSONArray: <https://static.javadoc.io/com.googlecode.json-simple/json-simple/1.1.1/org/json/simple/JSONArray.html>
4. Примеры использования JSON.simple: <https://www.baeldung.com/java-json-simple-jsonobject>

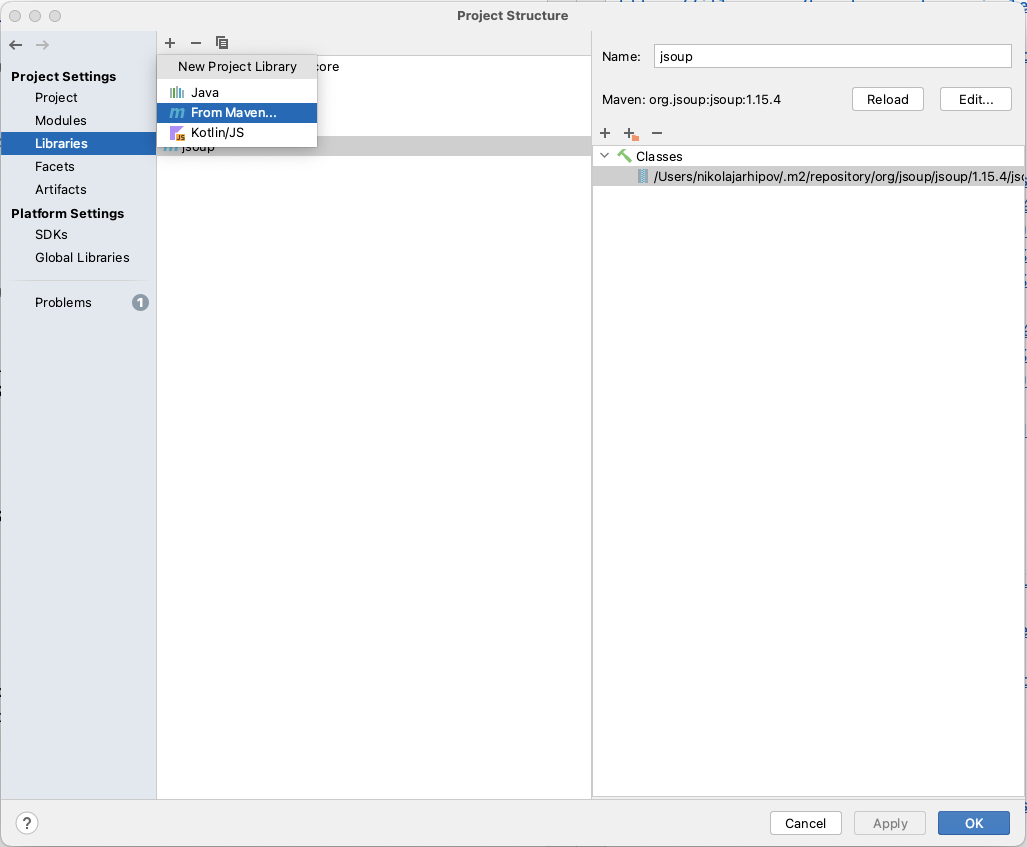
## 1.4 Пример работы с HTML

### 1.4.1 Как подключить библиотеку c помощью maven?

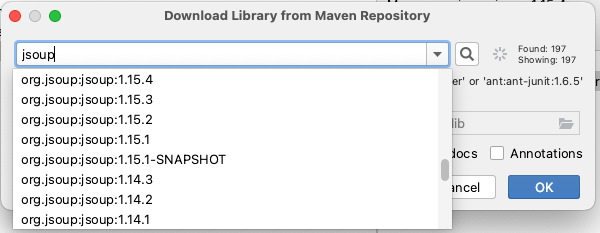
Для того чтобы добавьте библиотеку в IntelliJ IDEA в виде jar файла необходмо выбрать File – Project Structure, как показано на рисунке ниже.



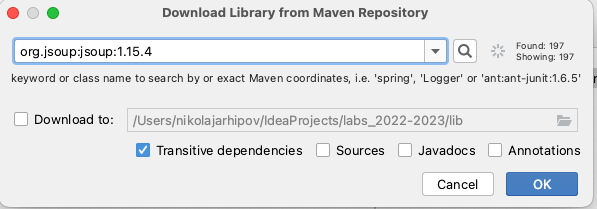
В окрывшемся окне нужно выбрать раздел Libraries, как показано на рисунке ниже.



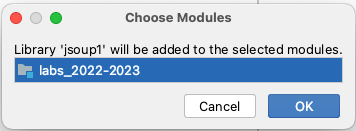
В открышемся окне в поиске введите jsoup и нажмите значок лупы для поиска.



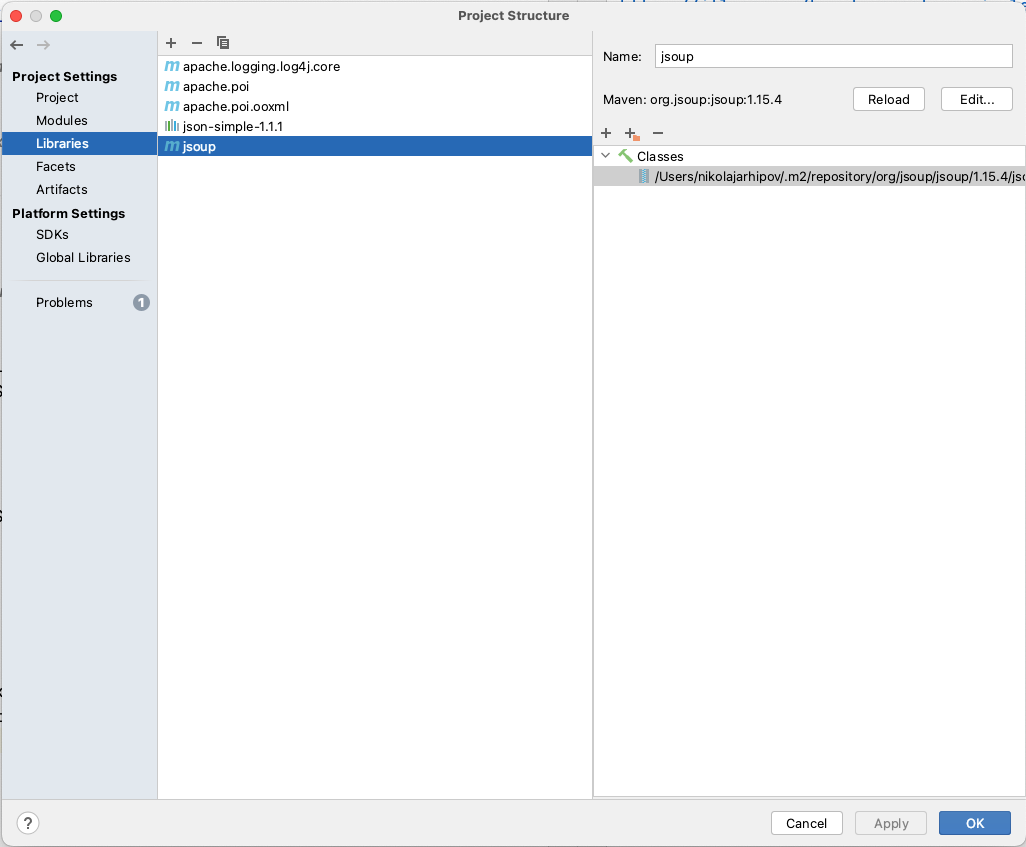
Выберите библиотеку как показано на рисунке ниже



Нажмите «ОК» как показано на рисунке ниже

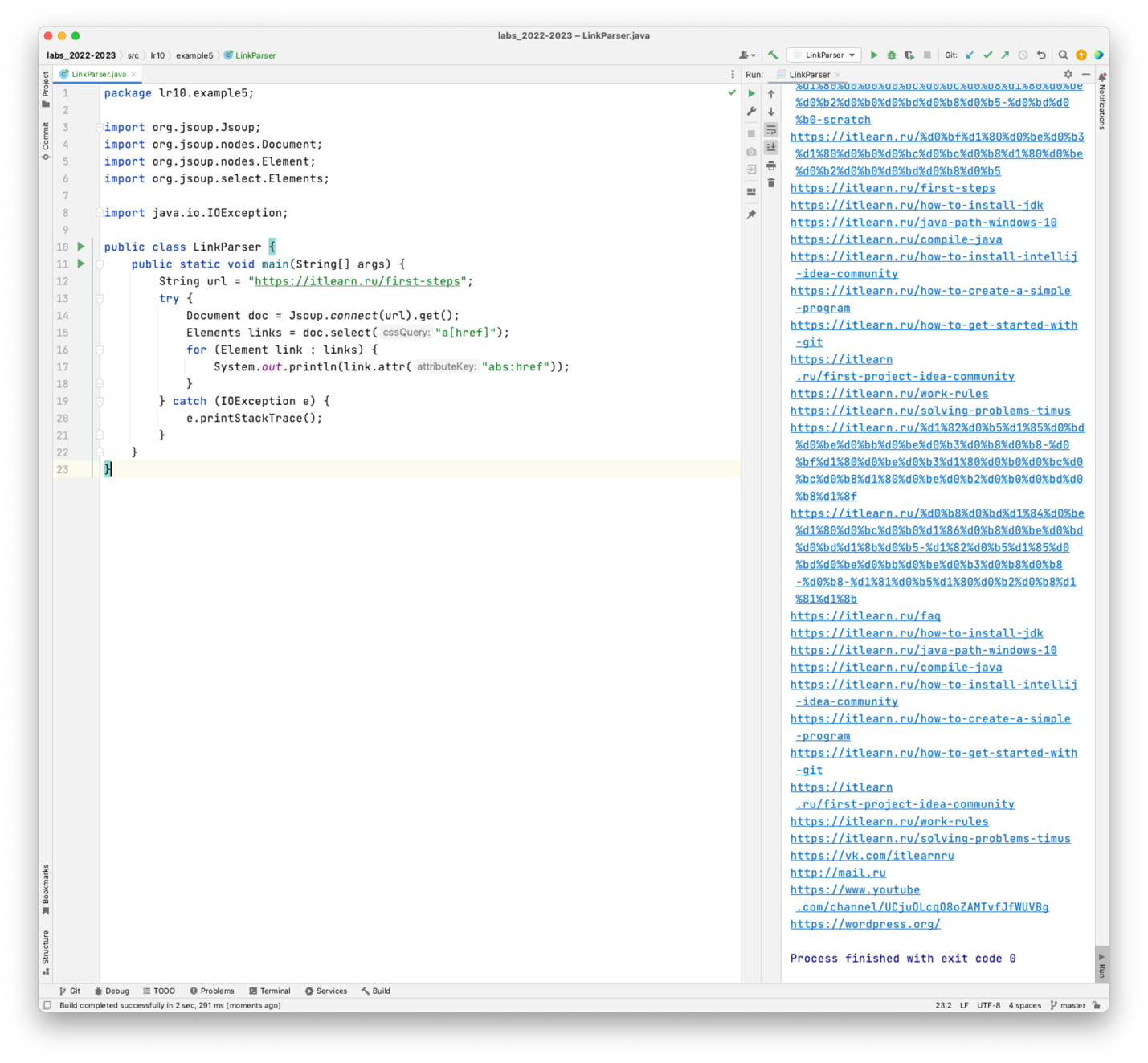


Должна появиться библиотека в списке библиотек, как показано на рисунке ниже.



### 1.4.2 Пример парсинга страницы <https://itlearn.ru/first-steps>

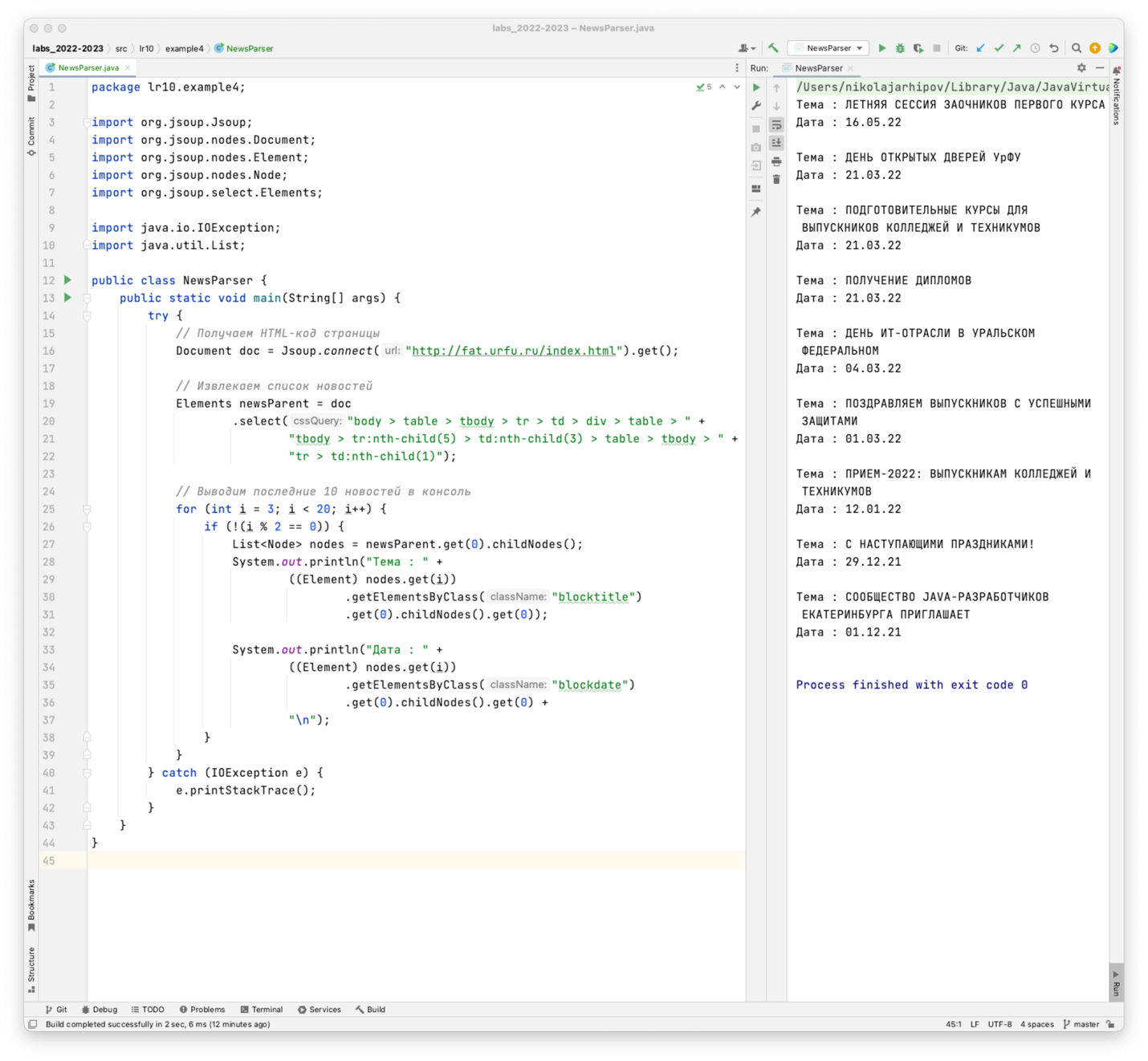
Код программы, который парсит страницу https://itlearn.ru/first-steps и выводит в консоль все ссылки на странице:



В этой программе мы используем библиотеку jsoup для подключения к сайту и получения его HTML-кода. Затем мы ищем все ссылки на странице, используя метод "select()", который принимает CSS-селектор, и выводим абсолютный URL каждой ссылки в консоль с помощью метода "attr()".

### 1.4.3 Пример парсинга страницы <http://fat.urfu.ru/index.html>

Код программы, который парсит страницу <http://fat.urfu.ru/index.html> и выводит в консоль темы новостей и даты их публикаций



Данная программа предназначена для парсинга (извлечения данных) с HTML-страницы, в данном случае - новостей с сайта fat.urfu.ru.

Сначала программа использует метод connect() из библиотеки Jsoup для получения HTML-кода страницы. Затем, с помощью метода select(), программа извлекает список новостей из HTML-кода.

Для каждой новости в списке программа выводит в консоль ее тему и дату. Для этого она применяет методы getElementsByClass() и childNodes() из библиотеки Jsoup.

### 1.4.4 Ссылки на документацию

Документация по Jsoup:

* Официальный сайт: <https://jsoup.org/>
* Документация: <https://jsoup.org/apidocs/>

Примеры работы с Jsoup:

* Статья на сайте Baeldung: <https://www.baeldung.com/java-with-jsoup>
* Руководство на сайте Vogella: <https://www.vogella.com/tutorials/jsoup/article.html>

## 1.5 Пример работы с Excel

Ниже приведен пример кода, который создает новую книгу Excel, создает новый лист в этой книге и записывает данные в ячейки:

### 1.5.1 Подключение библиотек c помощью maven

В разделе 1.4.1 «Как подключить библиотеку c помощью maven?» было показано как подключить библиотеку maven. Нужно основываясь на прошлом примере подключить следующие библиотеки:

apache.logging.log4j.core

org.apache.poi:poi:5.2.3

org.apache.poi:poi-ooxml:5.2.3

### 1.5.2 Как создать файл Excel



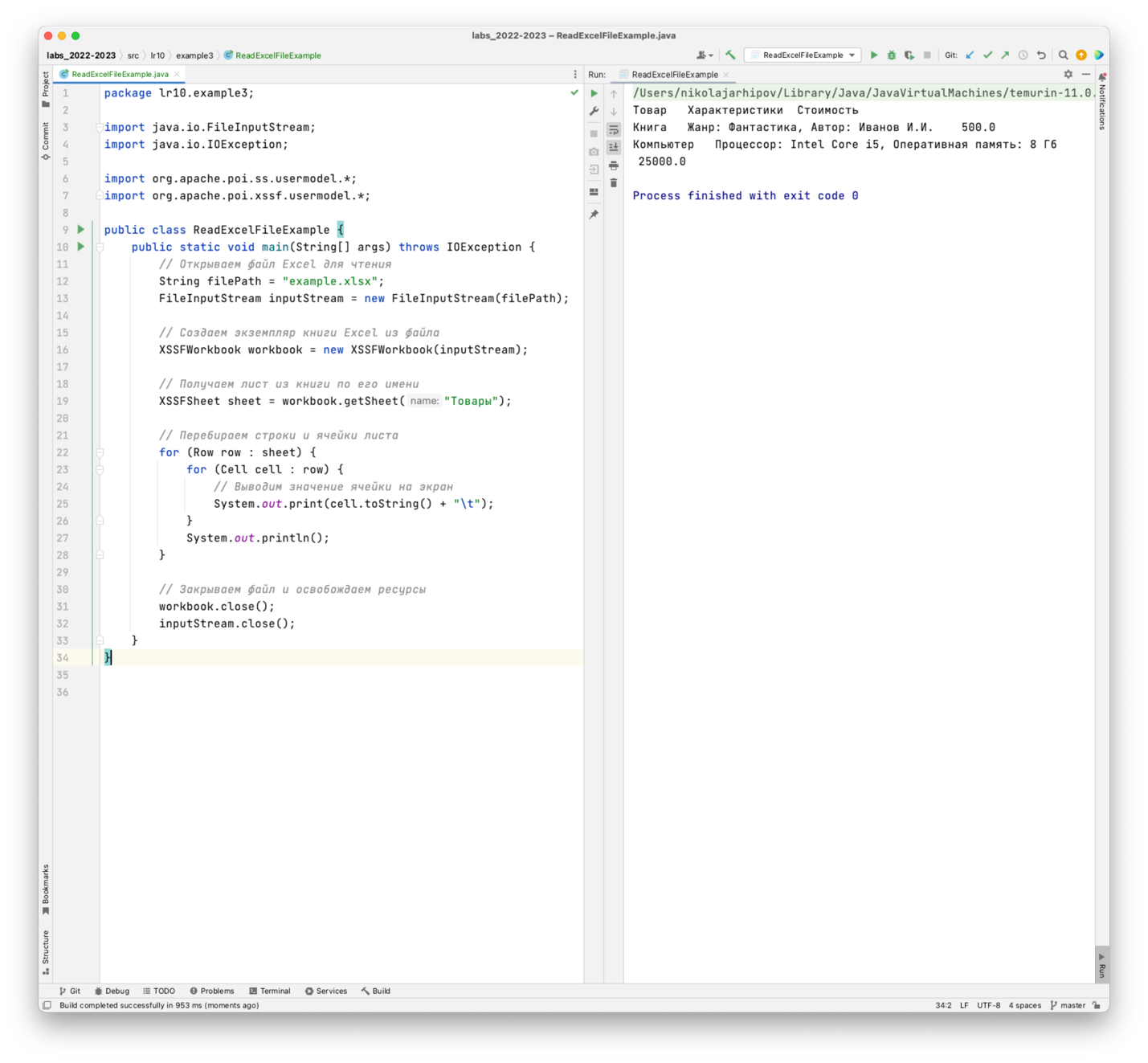
Эта программа создает новый Excel-файл в формате XLSX и записывает в него данные.

В начале программы мы импортируем необходимые классы из библиотеки Apache POI, которые позволяют работать с Excel-файлами. Затем мы определяем основной метод программы main, который выбрасывает исключение IOException, связанное с записью файла. Далее мы создаем новую книгу Excel, используя класс XSSFWorkbook. Затем мы создаем новый лист в книге с названием "Товары", используя метод createSheet. После этого мы заполняем ячейки листа данными. Сначала мы создаем заголовок таблицы, используя метод createRow. Затем мы создаем строки данных и заполняем их значениями, используя метод createRow. В этом примере мы создали две строки данных, содержащие информацию о книге и компьютере соответственно. Каждая строка содержит три ячейки: название товара, характеристики и стоимость.

После заполнения данных мы записываем книгу Excel в файл, используя объект FileOutputStream. Здесь мы записываем содержимое книги в файл с помощью метода write, а затем закрываем книгу и поток вывода.

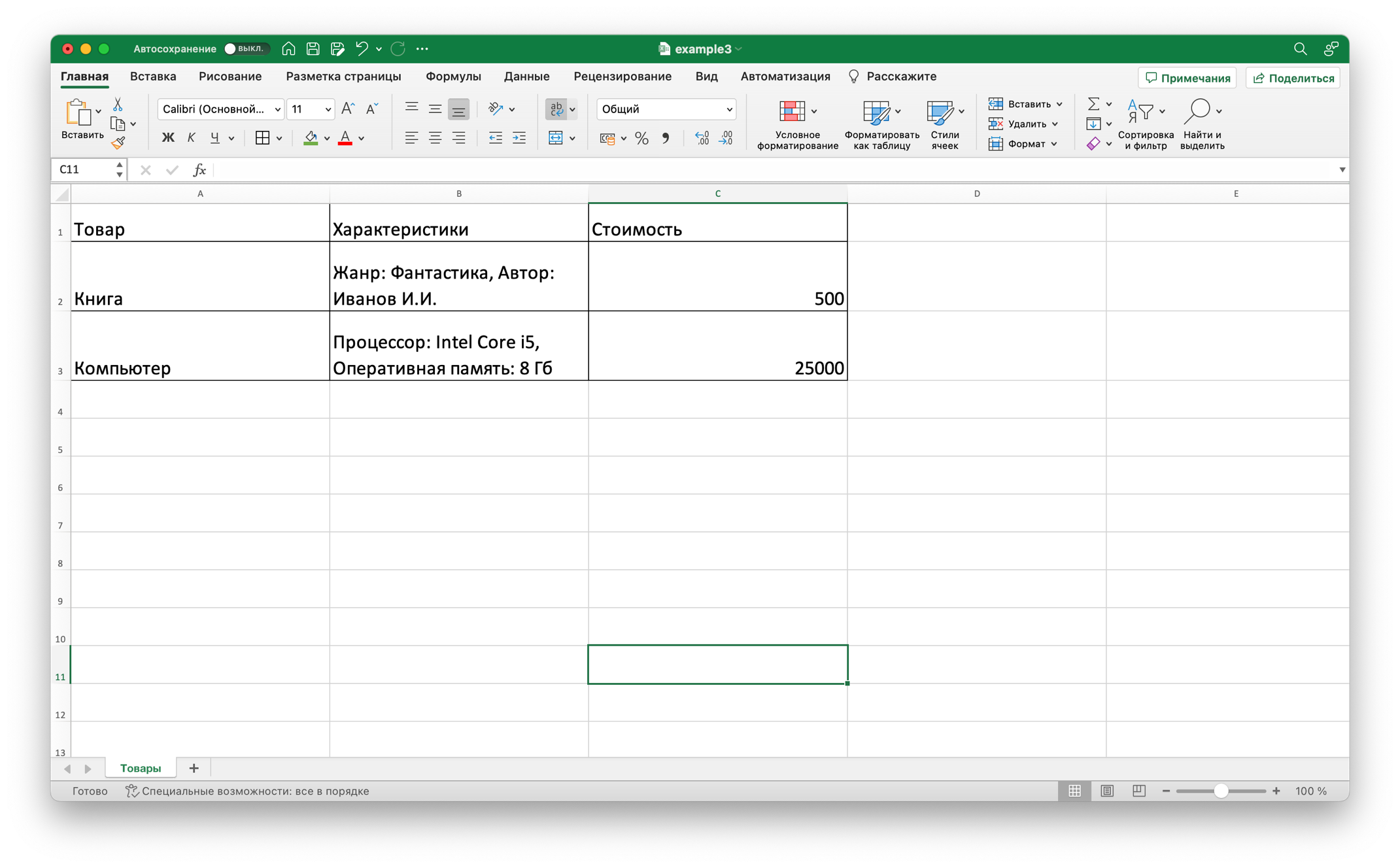
Наконец, программа выводит на экран сообщение о том, что данные успешно записаны в файл.

### 1.5.3 Как прочитать файл Excel



Эта программа является примером чтения данных из файла Excel в формате XLSX с использованием библиотеки Apache POI. В начале программы мы импортируем необходимые классы из библиотеки Apache POI, которые позволяют работать с Excel-файлами. Затем мы определяем основной метод программы **main**, который выбрасывает исключение **IOException**, связанное с чтением файла. Далее мы открываем файл Excel для чтения, используя объект **FileInputStream**. Затем мы создаем экземпляр книги Excel из файла, используя класс **XSSFWorkbook**. Далее мы получаем лист из книги по его имени, используя метод **getSheet**: После этого мы перебираем все строки и ячейки на листе, используя цикл **for**. Внутри цикла мы выводим на экран значение каждой ячейки с помощью метода **toString()**, который преобразует значение ячейки в строку. В конце программы мы закрываем файл и освобождаем ресурсы, используя методы **close()**: Таким образом, эта программа читает содержимое Excel-файла в формате XLSX и выводит его на экран.

### 1.5.4 Пример таблицы Excel



### 1.5.5 Ссылки на документацию

Документация Apache POI может помочь понять, как работают программы, которые используют эту библиотеку:

* Документация Apache POI: <https://poi.apache.org/documentation.html>

Кроме того, Java SE Documentation также может быть полезной для понимания базовых концепций Java:

* Java SE Documentation: <https://docs.oracle.com/en/java/javase/index.html>

# 2 ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

## 2.1 Выполните все примеры из раздела 1

## 2. 2 Выполните следующие доработки проекта XML парсера:

1. Сделайте в текстовом редакторе свой файл в формате XML в зависимости от варианта (Приложение 1).
2. Добавьте возможность записывать новые книги в XML-файл. Например, пользователь может ввести данные о новой книге, а программа добавит новый элемент **<book>** в XML-файл.
3. Добавьте возможность поиска книг по автору или году издания. Например, пользователь может ввести автора или год издания, а программа выведет список книг, удовлетворяющих этому критерию поиска.
4. Реализуйте функцию удаления книги из XML-файла. Например, пользователь может ввести название книги, которую хочет удалить, и программа удалит соответствующий элемент **<book>** из XML-файла.

### Подсказки:

1. Для записи данных в XML-файл можно воспользоваться классом **javax.xml.transform.Transformer**.

Пример:

Transformer transformer =

TransformerFactory.newInstance().newTransformer();  
DOMSource source = new DOMSource(doc);  
StreamResult result = new StreamResult(new File("src/lr10/example1/example.xml"));  
transformer.transform(source, result);

1. Для поиска книг по автору или году издания можно воспользоваться Java Stream API и методом **filter()**.

Пример:

List<Element> books = nodeList.stream()  
 .filter(node -> node.getNodeType() == Node.*ELEMENT\_NODE*)  
 .map(node -> (Element) node)  
 .filter(element -> {  
 String author = element.getElementsByTagName("author").item(0).getTextContent();  
 String year = element.getElementsByTagName("year").item(0).getTextContent();  
 return author.equalsIgnoreCase("Лев Толстой") && year.equals("1869");  
 })  
 .collect(Collectors.toList());

1. Для удаления элементов из XML-файла можно воспользоваться методами, предоставляемыми интерфейсом **org.w3c.dom.Node**. Например, для удаления элемента можно использовать следующий код:

Element bookElement = ...; *// получаем элемент книги,*

*// которую нужно удалить*Node parentNode = bookElement.getParentNode();  
parentNode.removeChild(bookElement);

## 2.2 Выполните следующие доработки проекта JSON парсера:

1. Сделайте в текстовом редакторе свой файл в формате JSON в зависимости от варианта (Приложение 1).
2. Добавьте функционал для поиска книг по автору.
3. Добавьте функционал для добавления новой книги в массив.
4. Добавьте функционал для удаления книги из массива по названию.

### Подсказки:

1. Для поиска книг по автору восользуетесь JSONArray.stream() для создания потока элементов массива и метод filter() для фильтрации элементов по заданному автору.

Пример:

JSONArray jsonArray = (JSONArray) jsonObject.get("books");  
String author = "Иванов";  
jsonArray.stream()  
 .filter(book -> book instanceof JSONObject)  
 .map(book -> (JSONObject) book)  
 .filter(book -> author.equals(book.get("author")))  
 .forEach(book -> {  
 System.out.println("\nТекущий элемент: book");  
 System.out.println("Название книги: " + book.get("title"));  
 System.out.println("Автор: " + book.get("author"));  
 System.out.println("Год издания: " + book.get("year"));  
 });

1. Для добавления новой книги в массив необходимо использовать метод JSONArray.add() для добавления нового объекта JSONObject в массив JSONArray.

Пример:

JSONArray jsonArray = (JSONArray) jsonObject.get("books");  
JSONObject newBook = new JSONObject();  
newBook.put("title", "Новая книга");  
 newBook.put("author", "Новый автор");  
 newBook.put("year", 2023);  
 jsonArray.add(newBook);

1. Для удаления книги из массива необходимо использовать метод Iterator.remove() для удаления объекта из массива JSONArray.

Пример:

JSONArray jsonArray = (JSONArray) jsonObject.get("books");  
String title = "Название книги";  
Iterator iterator = jsonArray.iterator();  
while (iterator.hasNext()) {  
 JSONObject book = (JSONObject) iterator.next();  
 if (title.equals(book.get("title"))) {  
 iterator.remove();  
 }}

## 2.3 Выполните следующие доработки проекта HTML парсера

Доработка парсера новостей:

1. Добавьте функционал записи полученных данных в файл для сохранения информации на будущее.
2. Добавьте обработку ошибок при получении HTML-кода страницы, например вывод сообщения об ошибке и попытка переподключения к сайту.

## 2.4 Выполните следующую доработки проекта Excel парсера

1. Улучшение работы с ошибками. Программы, которые читают Excel файлы, могут столкнуться с ошибками, такими как неправильный формат файла, отсутствие запрашиваемого листа и т.д. Улучшение работы с ошибками может включать более подробные сообщения об ошибках, рекомендации для исправления ошибок и возможность повторного запуска программы после исправления ошибок.

Приложение 1

1. Книги (Book)

2. Покупки/продукты (Product)

3. Задачи (Task)

4. Дни рождений людей (Birthday)

5. Домашние животные (Animals)

6. Предметы в университете (Lesson)

7. Автомобили (Automobile)

8. Сотрудники (Employee)

9. Фильмы (Movies)

10. Музыкальные произведений (Songs)

# 3 ОПИСАНИЕ РЕЗУЛЬТАТА ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

Результат выполнения ЛР:

1) Отчет о выполнении лабораторной работы приложенный к заданию, содержащий ссылку на репозиторий github.com;

В отчете по лабораторной работе должны быть представлены:

1. Титульный лист;

2. Цель работы;

3. Описание задачи;

4. Ход выполнения (содержит код программы);

5. Вывод;

Оформление:

а) шрифт Times New Roman;

б) размер шрифта 12 или 14;

в) межстрочный интервал 1,5.

Отчет выполняется индивидуально.