**Лабораторная работа 7**

**Библиотека jsoup**

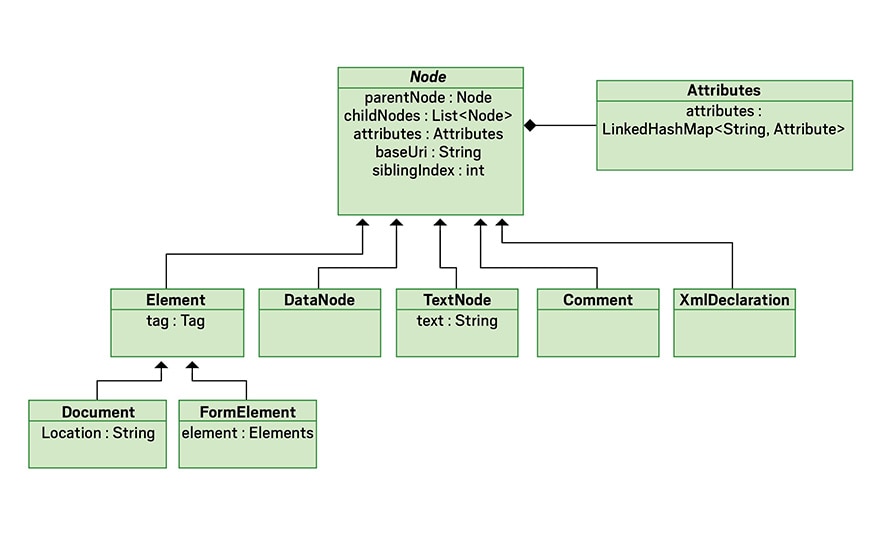
Абстрактный класс org.jsoup.nodes.Node лежит в основе библиотеки jsoup (диаграмма классов представлена на рис. 1).

Рисунок 1. Диаграмма классов Jsoup

Node – узел в дереве DOM, который может быть документом, текстовым узлом, комментарием или элементом формы в документе. Класс Node знает свой родительский узел и все свои дочерние узлы.

Класс Element представляет элемент HTML и состоит из имени тега, атрибутов и дочерних узлов.

Класс Attributes представляет собой контейнер для атрибутов HTML-элементов и входит в состав класса Node.

**Добавление зависимости в pom.xml**

<dependency>

<groupId>org.jsoup</groupId>

<artifactId>jsoup</artifactId>

<version>Версия</version>

</dependency>

Класс org.jsoup.Jsoup предоставляет базовые методы, которые могут анализировать HTML-документ, переданный в виде:

* файла,
* строки,
* HTML-документа, предоставленного через URL.

**Загрузка HTML страницы**

Веб-страницу можно загрузить:

1. из файла

File input = new File("/input.html");

Document doc = Jsoup.parse(input, "UTF-8", "http://example.com/");

1. из текста

String html = "<html><head><title>First parse</title></head> <body><p>Hello, world!</p></body></html>";

Document doc = Jsoup.parse(html);

1. из Интернета.

Document doc = Jsoup.connect("http://github.com/").get();

При создании документа можно указать userAgent и timeout.

Document doc = Jsoup.connect(request)

.userAgent( "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:36.0) Gecko/20100101 Firefox/36.0")

.timeout(5000).get();

При открытии сайта браузер посылает HTTP-запрос к серверу, который занимается обслуживанием этого сайта, дожидается ответа от сервера и обрабатывает его определенным образом. HTTP-запрос содержит в себе набор информации, из которой выделяют метод обработки данных (GET, POST), адрес запрашиваемого ресурса, IP-адрес отправителя и User-Agent.

User-Agent – это текстовая часть запроса, которую веб-приложение использует для сообщения сайту информации о себе и корректного отображения страниц.

User-Agent браузера содержит название и версию приложения, а так же данные об операционной системе компьютера: версия, разрядность, язык по умолчанию и другие параметры. Помимо браузеров User-Agent есть у поисковых роботов. В них содержится имя робота и ссылка на ресурс, регламентирующий его работу.

Чтобы определить User-Agent, нужно в адресной строке браузера набрать команду about: или browser://version.

**Базовые операции библиотеки Jsoup**

**Способы выбора элементов DOM**

Способы перебора проанализированных HTML-элементов и поиска запрошенных:

1. методы getElementBy\*, специфичные для DOM,
2. селекторы CSS и jQuery.

**Методы поиска элементов**

|  |  |
| --- | --- |
| Метод | Описание |
| Element getElementById(String id) | Поиск элемента по ID, с данного места в документе или ниже данного элемента. |
| Elements getElementsByTag(String tag) | Поиск элементов с определенным названием тега, включая и рекурсивно внизу данного элемента. |
| Elements getElementsByClass(String className) | Поиск элементов с параметром classNam, включая или внизу данного элемента. |
| Elements getElementsByAttribute(String key) | Поиск элементов с атрибутами, не отличая прописные и строчные буквы. |
| Elements siblingElements() | Возвращает братские элементы текущего элемента. |
| Element firstElementSibling() | Возвращает первый братский элемент текущего элемента. |
| Element lastElementSibling() | Возвращает последний братский элемент текущего элемента. |

**Методы получения данных на Element.**

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| String attr(String key) | Возвращает значение атрибута по данному key этого элемента. |
| void attr(String key, String value) | Настроить значение атрибута. Если атрибут существует, он будет заменен. |
| String id() | Возвращает атрибут ID, если есть, или возвращает пустой string если нет. |
| String className() | Возвращает строку значения атрибута "class", может содержать много class name, отделенныех пробелами. (Например<div class="header gray"> возвращает "header gray") |
| Set<String> classNames() | Возвращает все class names.  Например <div class="header gray">, возвращает набор из 2-х элементов "header" и "gray". |
| String text() | Возвращает сочетание текста и всех текстов подэлементов. |
| void text(String value) | Настраивает текст для данного элемента. |
| String html() | Возвращает String в HTML внутри данного тега.  Например ***<div><p>a</p></div>*** возвращает ***<p>a</p>***.  (**Node.outerHtml()** возвратит ***<div><p>a</p></div>***.) |
| void html(String value) | Настраивает Html внутри данного элемента. Удалить все готовые HTML внутри. |
| Tag tag() | Возвращает Tag этого элемента. |
| String tagName() | Возвращает название тега данного элемента. Например, div. |

**Пример 1.** Вывести названия всех тегов и текст**.**

static String htmlText = "<!DOCTYPE html>" +

" <html>" +

" <head>" +

" <title>Java Magazine</title>" +

" </head>" +

" <body>" +

" <h1>Hello World!</h1>" +

" </body>" +

"</html>";

Document document = Jsoup.parse(htmlText);

Elements allElements = document.getAllElements();

for (Element element : allElements) {

System.out.println(element.nodeName()+ " " + element.ownText());

}

Вывод на экран.

#document

html

head

title Java Magazine

body

h1 Hello World!

**Пример 2.** Извлечь из страницы все ссылки с тегами <a>, помеченные как "New!". Была сделана маркировка с помощью тега <font> с текстом New! рядом со ссылкой.



Elements allElements = document.getElementsByTag("a");

for (Element element : allElements) {

if ("New!".equals(element.nextElementSibling()!=null

? element.nextElementSibling().ownText()

: "")) {

System.out.println(element.ownText());

}

}

Альтернативное извлечение тегов с помощью селекоров.

Elements allElements = document.select ("a[href\*=#]");

Комбинация селекторов для уточнения отбора. Выбор тега <font>, содержащего текст new! и находящегося после ссылки с href, ссылка начинается со значения #.

Elements allElements = document.select

("a[href\*=#] ~ font:containsOwn(New!)");

for (Element elem : allElements) {

System.out.println(elem.previousElementSibling().ownText());

}

Селекторы находят тег <font> в качестве элемента. Затем у этого элемента вызывается метод previousElementSibling(), чтобы шагнуть на один элемент назад к ссылке.

Метод select () доступен в классах Document, Element и Elements.

**Обход узлов**

jsoup предоставляет интерфейс org.jsoup.select.NodeVisitor, который содержит два метода: head () и tail (). Реализация анонимного класса этого интерфейса и передача его в качестве параметра в метод document.traverse() позволяет получить обратный вызов при первом и последнем посещении узла.

**Пример 3.** Использовать этот метод для обхода простого HTML-текста и вывода всех деталей узла.

document.traverse(new NodeVisitor() {

public void head(Node node, int depth){

System.out.println("Node start: " + node.nodeName());

}

public void tail(Node node, int depth){

System.out.println("Node end: " + node.nodeName());

}

});

Дана строка

static String htmlText = "<!DOCTYPE html>" +

"<html>" +

"<head>" +

"<title>Java Magazine</title>" +

"</head>" +

"<body>" +

"<h1>Hello World!</h1>" +

"</body>" +

"</html>";

Результат:

Node start: #document

Node start: #doctype

Node end: #doctype

Node start: html

Node start: head

Node start: title

Node start: #text

Node end: #text

Node end: title

Node end: head

Node start: body

Node start: h1

Node start: #text

Node end: #text

Node end: h1

Node end: body

Node end: html

Node end: #document

**Обработка URL**

При обработке веб-страниц ссылки обычно являются не абсолютными, а относительными. Добавление параметра abs: возвращает полный путь для URL (абсолютный путь).

Document doc = Jsoup.connect("http://jsoup.org").get();

Element link = doc.select("a").first();

String relHref = link.attr("href"); // "/"

String absHref = link.attr("abs:href"); // "http://jsoup.org/";

**Пример 4.** Поиск прямых наследников элемента.

String html ="<table id='tabletext'>" +

"<tr align='center' style='background:#FFFFFF'>" +

"<td>10</td>" +

"<td>10.00</td>" +

"<td>&pound;0.00</td>" +

"<td>&pound;280.00</td>" +

"<td>" +

"<a href='/cart.php?action=add&qty=10&id=2628' title='Click here to add this item to your cart'>" +

"<img alt='Click here to add this item to your cart' src='/images/addtocart.gif' border='0' />" +

"</a>" +

"</td>" +

"</tr>" +

"</table>";

//все столбцы таблицы

Elements elements = doc.select("#tabletext > tbody > tr > td"); Element e = elements.get(3); // 4-ый столбец

**Селекторы в jsoup**

Элементы **JSoup** поддерживают синтаксис похожий на **CSS** (или **JQuery**), помогающий найти соответствующие элементы. Методы выбора есть в классах **Document**, **Element** и **Elements**.

**Selector**

|  |  |
| --- | --- |
| **Селектор** | **Описание** |
| Tagname | Искать элементы по тегу.  Например **a** |
| ns|tag | Искать элементы по тегу в пространстве имен (namespace).  Например: **fb|name** значит найти элементы **<fb:name>** |
| #id | Искать элементы по ID.  Например **#logo** |
| .class: | Искать элементы по названию класса.  Например **.masthead** |
| [attribute] | Элементы с атрибутами.  Например **[href]** |
| [^attr] | Элементы с атрибутами с приставкой.  Например **[^data-]** искать элементы с атрибутами начинающимися на data- |
| [attr=value] | Элементы со значениями атрибута.  Например, **[width=500]** (Можно использовать кавычки) |
| [attr^=value], [attr$=value], [attr\*=value] | Элементы со значениями атрибута  начинающиеся,  заканчивающиеся,  содержащие значение.  Например, **[href\*=/path/]** |
| [attr~=regex] | Элементы со значениями, совпадающими с регулярным выражением.  Например, **img[src~=(?i)\.(png|jpe?g)]** |
| \* | Все элементы. |

**Комбинации Selector**

|  |  |
| --- | --- |
| **Selector** | **Описание** |
| el#id | Элементы с ID.  Например **div#logo** |
| el.class | Элементы с классом.  Например **div.masthead** |
| el[attr] | Элементы с атрибутом.  Например **a[href]** |
| el[attr].class | Элементы с атрибутом и классом  Например **a[href].highlight** |
| ancestor child | Родительский элемент – наследованный элемент  Подэлементы родительского элемента.  Например .**body p** ищет элемент **p** везде под блоком с классом **body** |
| parent > child | Прямые наследники родительского элемента.  Например  **div.content > p** найдет элементы **p**, которые являются прямыми наследниками div, имеющего **class ='content'**;  **body > \*** найдет прямые подэлементы тега body |
| siblingA + siblingB | Найти элементы братья B сразу перед элементом A.  Например **div.head + div** |
| siblingA ~ siblingX | Найти элементы братья X перед элементом A.  Например **h1 ~ p** |
| el, el, el | Группа с разными **Selector**, ищет элементы подходящие к одному из **Selector**.  Например **div.masthead**, **div.logo** |

**Pseudo selectors**

|  |  |
| --- | --- |
| **Selector** | **Описание** |
| :lt(n) | Поиск родственных элементов с индексом меньше n.  Например **td:lt(3)** |
| :gt(n) | Поиск родственных элементов с индексом больше n.  Например **div p:gt(2)** |
| :eq(n) | Поиск родственных элементов с индексом равным n. Например form **input:eq(1)** |
| :has(seletor) | Поиск элементов, содержащих элементы, совпадающие с selector.  Например **div:has(p)** |
| :not(selector) | Поиск элементов несовпадающих с **Selector**.  Например **div:not(.logo)** |
| :contains(text) | Поиск элементов содержащих данный текст. Регистр не важнен.  Например **p:contains(jsoup)** |
| :containsOwn(text) | Поиск элементов, которые напрямую содержат данный текст |
| :matches(regex) | Поиск элементов, где текст не совпадает с определенным выражением.  Например **div:matches((?i)login)** |
| :matchesOwn(regex) | Поиск элементов, где текст совпадает с определенным выражением. |
| Примечание: Индексы в pseudo начинается с 0. | |

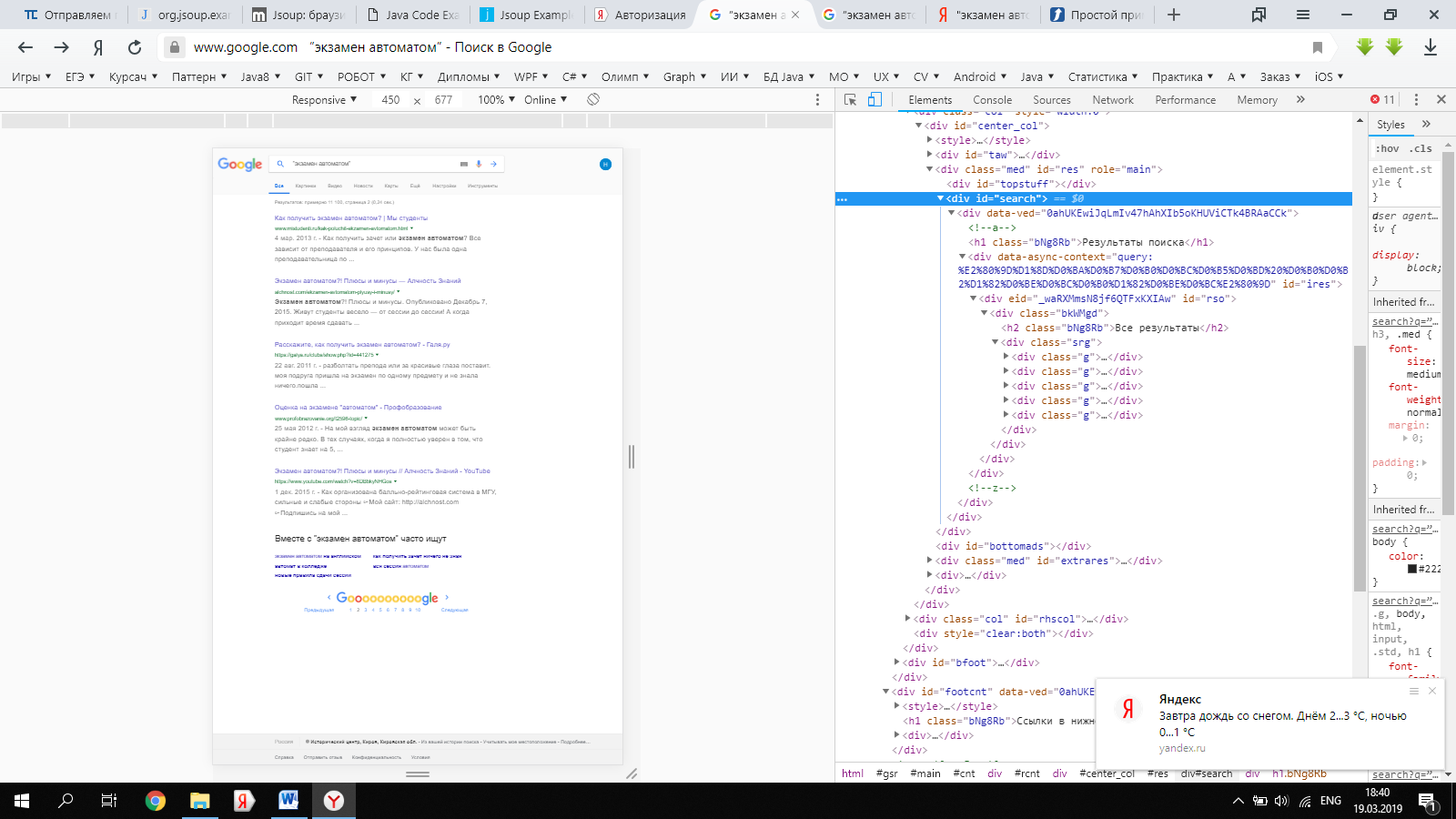
**Самостоятельная работа. Задачи**

**Задача 1**. Отправить поисковый запрос в Google. Введите в адресную строку Google get-запрос: https://www.google.com/search?q=”selenium”&num=5

Ответом должна быть страница со ссылками.

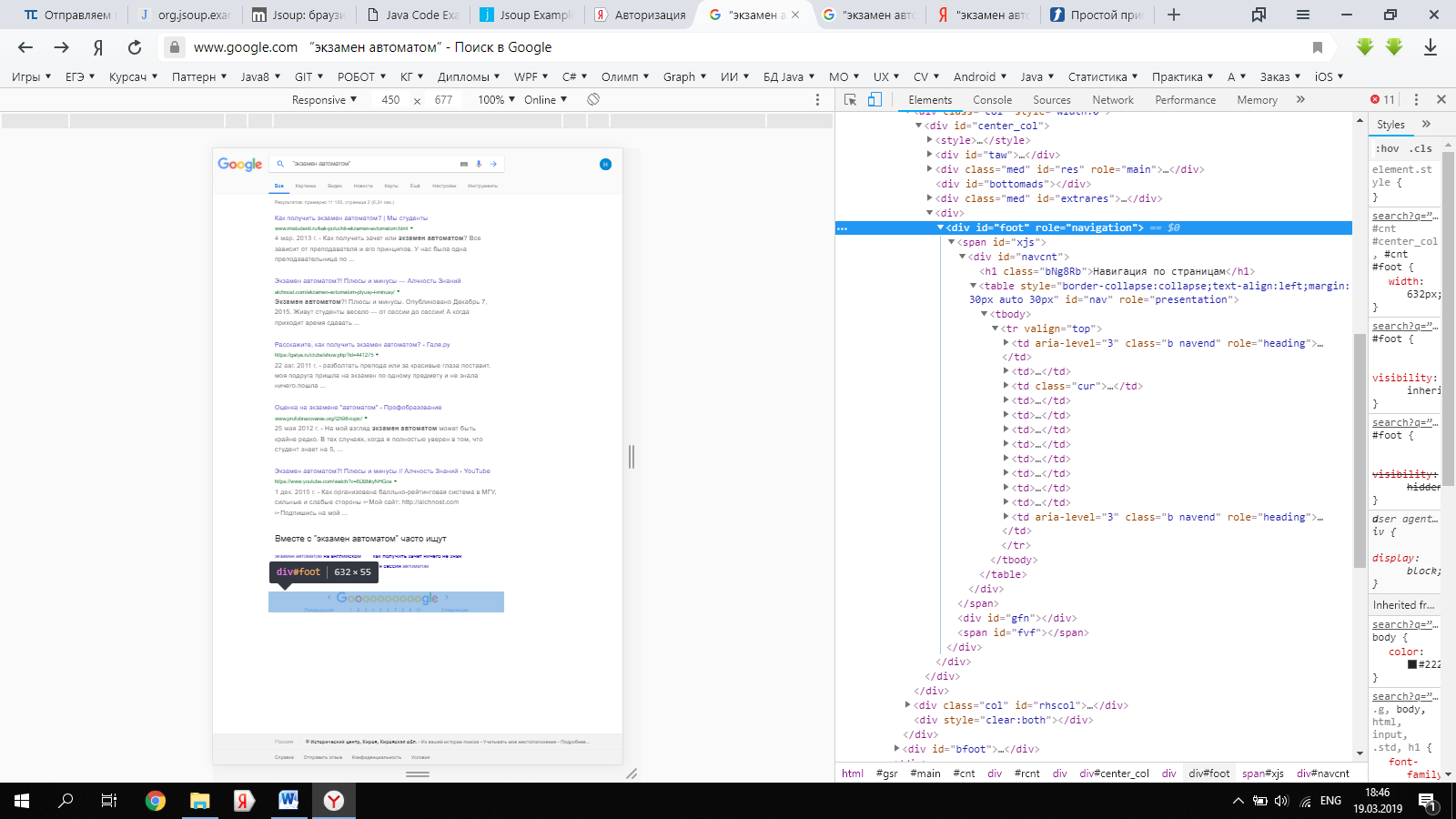
Напишите метод, принимающий 2 параметра (фразу поиска и количество ссылок на странице) и возвращающий список ссылок, которые были найдены.

Результаты поиска – ссылки можно найти в элементе div c id="search".



Переход по страницам можно найти в div с id="foot"

Осуществите переход на страницу с каким-то номером.



**Задача 2**. Задание по вариантам:

1. Соберите новости с Яндекс.Новости.

2. Соберите фильмы с сайта Кинопоиск.

Список полезных ссылок:

<https://jsoup.org/cookbook/>

<https://www.javacodeexamples.com/jsoup-examples>

**Задача 3**. Выполните поиск картинок. Используя пример из <https://www.javacodeexamples.com/jsoup-download-images-from-webpage-example/815>, скачайте несколько картинок в папку src/main/resources/images (эта папка должна существовать на диске).

Можно скачать картинки со ссылок предыдущего задания.

Задача 4. Скачайте список номеров автобусов с сайт https://cdsvyatka.com. По номеру автобуса найдите список остановок. Используйте Selenium WebDriver.