**Обзор библиотек для работы с XML файлами**

**DOM**

1. Представляет xml документ в виде дерева (вложенность объектов друг в друга);
2. Низкая скорость работы;
3. Неэффективен по используемой памяти;
4. Удобен и прост в использовании.

**SAX**

1. Обеспечивает низкоуровневый доступ к xml файлу: файл читается построчно по xml-событиям, причём обработчики событий необходимо создавать самим;
2. Эффективен по скорости работы;
3. Эффективен по используемой памяти;
4. Сложен в использовании по сравнении с DOM.

**StAX**

1. Модификация SAX: считывает файл построчно по xml-событиям, но уже способен самостоятельно их обрабатывать;
2. Эффективен по скорости работы;
3. Эффективен по используемой памяти;
4. Более прост в использовании по сравнению с SAX, но более сложен по сравнению с DOM.

**XPath**

1. Язык запросов xml. Его можно сравнить с SQL (декларативный язык), в котором мы говорим, ЧТО нам нужно получить, а не КАК нам нужно это получить;
2. В соответствии с пунктом 1, XPath предоставляет доступ к какому-то определённому элементу, то есть не нужно последовательно идти по xml-файлу, а можно сразу получить то, что нужно;
3. Прост в использовании.

Для выполнения задания были взяты DOM (для записи) и StAX (для чтения). Они были взяты, потому что отличаются по принципу чтения и представления xml-файла, чтобы изучить их работу и сравнить сложность работы с ними. Был взят StAX, а не SAX, так как он удобнее в использовании. XPath не был взят, так как он используется для выборочного чтения xml-файла, а в нашем задании нужно читать весь файл, чтобы конвертировать его в формат json и обратно.