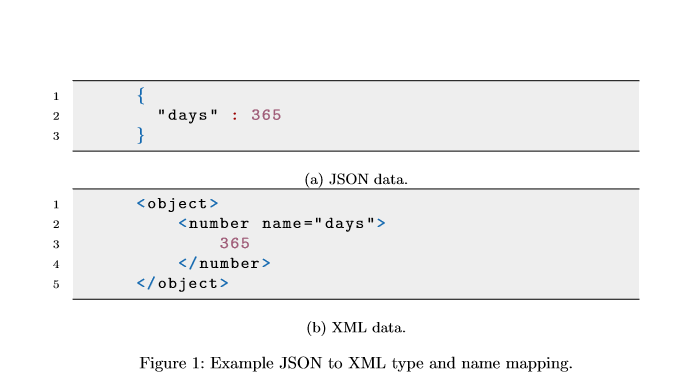
**1 Introduction**

This assignment entails creating a simple program that will allow users to give arbitrary JSON data and have it converted to XML according to the following standard.

In order to complete this assignment you must perform the following tasks.

* Fully implement the Java interface XMLJSONConverterI according to the given speciﬁcation.
* Provide a body for the the method createXMLJSONConverter in the class ConverterFactory.
* Provide a build script of some type to build you application.
* Provide a simple command line interface for running your program. Each of the aforementioned items is elaborated in the following sections. In general, the more you treat this project as though it were going to be real production code the better. You program MUST output valid XML!

**2 JSON to XML Speciﬁcation**

**2.1 Overview**

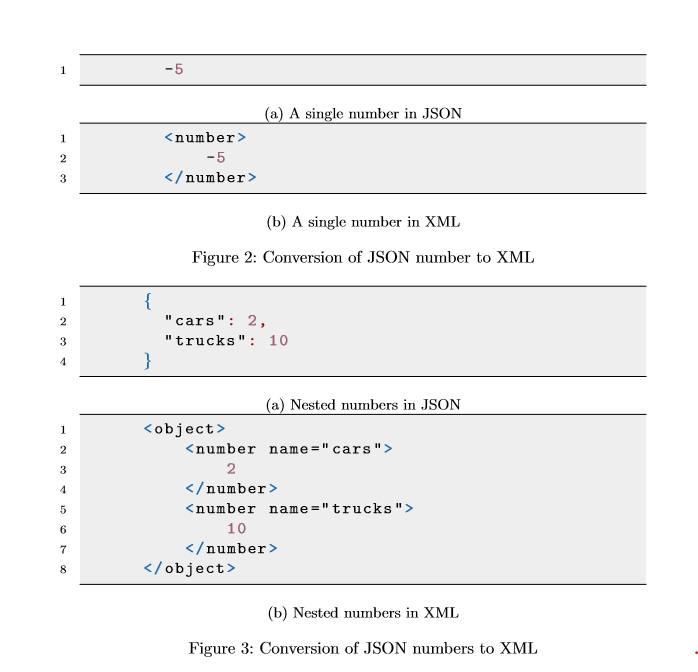
JSON does not strictly map to XML in a single way. There are many diﬀerent ways one could create a mapping from a JSON document to a XML document. This section describes the mapping that your code MUST follow in order to be considered valid.

**2.2 General Constraints**

In JSON objects you are given name value pairs. This can be mapped to XML in several ways. For this program, the XML element name corresponds to the type of the element, not the name of the element. The name of the JSON element in a JSON object is given as an attribute on the XML element. Only JSON array and object values may be at the top level of a ﬁle. The speciﬁcs of these values are discussed in greater detail in later sections.

**2.3 JSON Types**

JSON supports the following types, all of which must be mapped into your XML output. Only JSON objects and arrays are supported as top level values.



**2.3.1 Number**

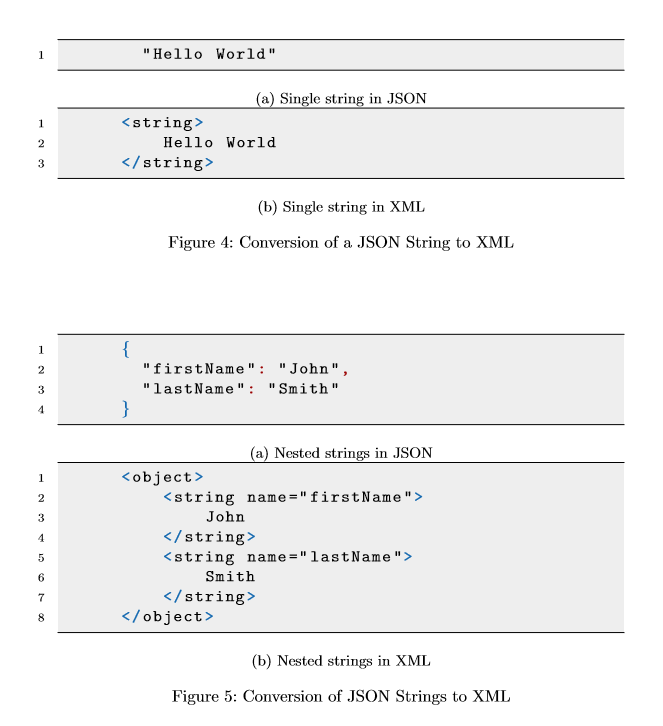
A JSON element with a number as a value should map to an XML element named <number>, with the number as the single and only value between the opening and closing tags. An example of this is shown in ﬁgure 2 and 3.

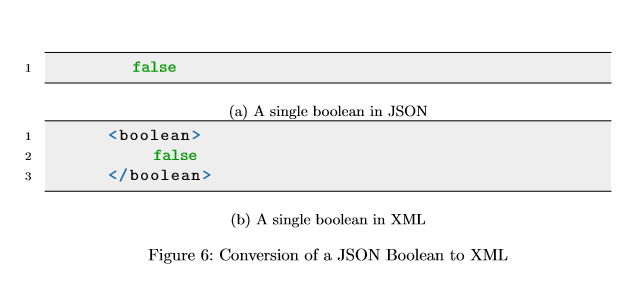
**2.3.2 String**

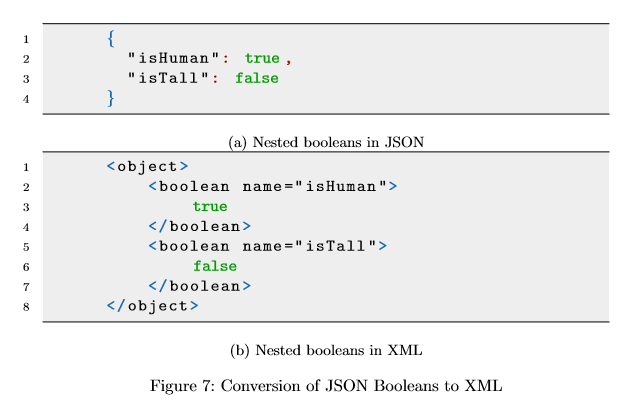
A JSON element with a string as a value should map to an XML element named <string>, with the string as the single and only value between the opening and closing tags. An example of this is shown in ﬁgure 4 and 5.

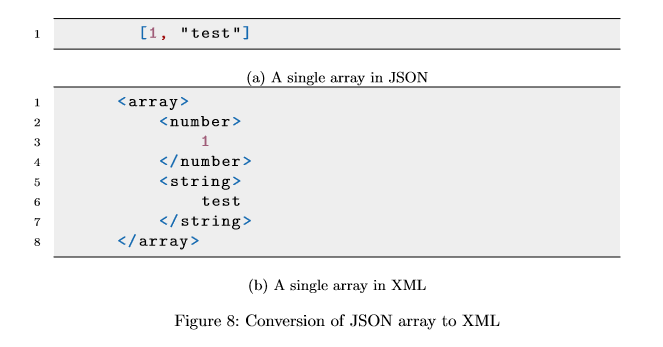
**2.3.3 Boolean**

A JSON element with a Boolean as a value should map to an XML element named <boolean>, with the boolean as the single and only value between the opening and closing tags. The only valid boolean values are true and false.An example of this is shown in ﬁgure 6 and 7.







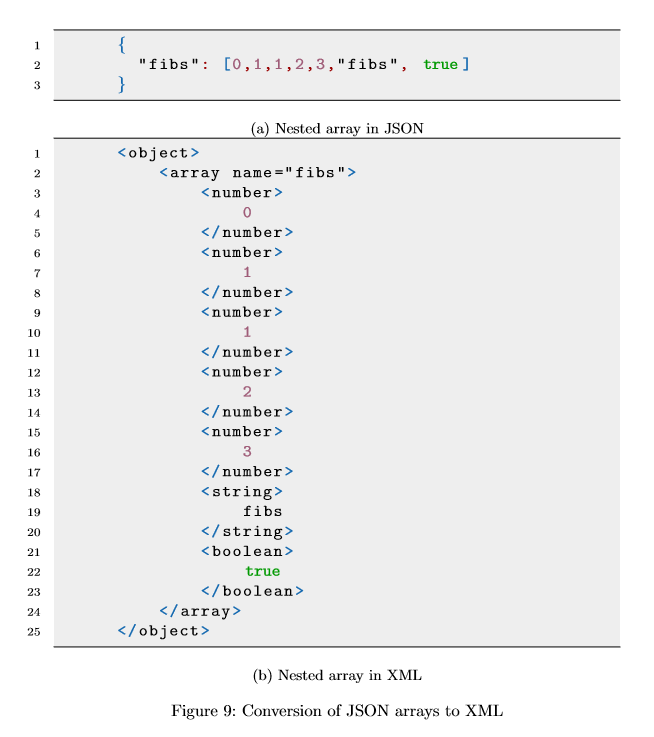


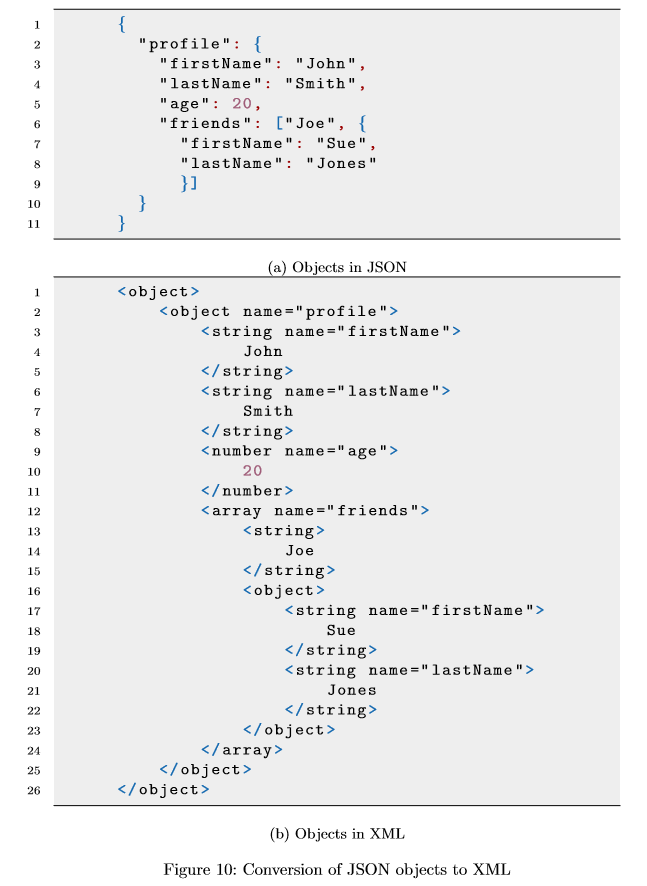
**2.3.4 Array**

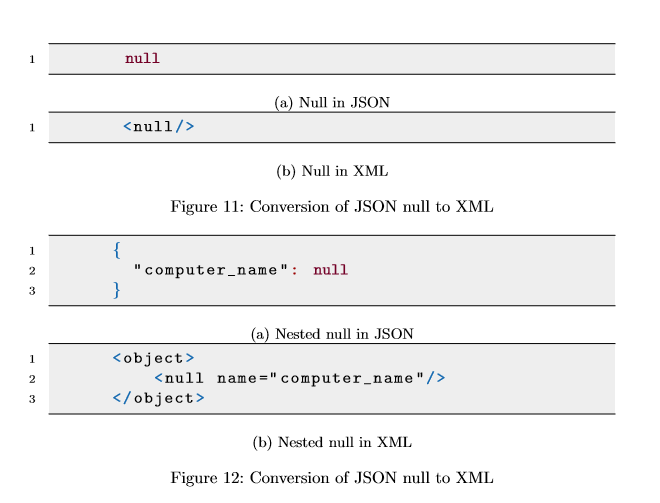
A JSON element with an array as a value should map to an XML element named <array>. For each element in the JSON array there should be a corresponding XML sub-element denoting the value and type. A JSON array may contain arbitrary sub values. type. Elements of the array should not have a name attribute. Remember, JSON arrays are not required to be homogeneous! An example of this is shown in ﬁgure 8 and 9.

**2.3.5 Object**

A JSON element with an object as a value should map to an XML element named <object>. Objects can contain arbitrary sub-values.An example of this is shown in ﬁgure 10. Objects are the only JSON types that generate name attributes on XML tags.







**2.3.6 null**

A JSON element with an null as a value should map to an XML element named <null/>. Unlike other XML representations for JSON values, a null value does not have a closing tag, rather it is self-closing by adding a forward slash before the end of the element tag. An example of this is shown in ﬁgure 11 and 12.

**2.3.7 Examples**

Besides the examples shown here, there examples ﬁles in the examples/ folder in the archive with which you were provided. These were generated with our own reference implementation. If anything in those ﬁles seems to not match up with this document, please contact us and let us know.

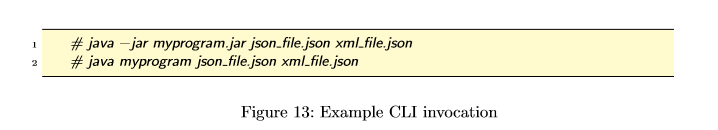
**3 Java Code**

**3.1 Overview**

There are very few constraints on what you may and may not do in this project. The few that do exist are listed below. Otherwise feel free to do whatever you think is best to accomplish the task. We will be providing you a few basic Java ﬁles from which to build the project. Their use and purpose is described below.

**3.2 Open Source Libraries**

You may use any open source libraries to help you accomplish this project without any restrictions, however You must either include these libraries in your submitted code, or have them automatically downloaded through your build tool.



**3.3 XMLJSONConverterI**

The core of the program you will implement revolves around the interface XMLJSONConverterI. This interface deﬁnes a single method convertJSONtoXML which takes two File values, an input JSON and an output XML. Your code must convert the given JSON to XML based on the speciﬁcation in section 2.

**3.4 ConverterFactory**

The ConverterFactory class is a Factory which creates instances of XMLJSONCovnerterI. You should complete the method stub found in that ﬁle and have it return an instance of your class implementing XMLJSONConverterI. It is required so that we can test your code without knowing your implementation. You must not change the method signature for this method at all.

**3.5 CLI**

You program must have a very simple command line interface. The ﬁrst argument is taken to be the path to the input JSON ﬁle and the second argument is taken to be the path to the XML output ﬁle. Any other number of arguments should print an usage message. The XML ﬁle may or may not exist, but the path must be valid. Example invocation can be seen in Figure 13.

**4 Building**

**4.1 Java Version**

You code must compile on a Linux system running Java 7. Although it should not matter, I will be testing your code on a Debian 7 system running the openjdk-7-jdk package.

**4.2 Build Script**

You must provide some sort of build script for making your project. This can be Ant, Make, SBT, a simple Shell Script, or anything else. The only requirement is that I have to be able to install the build tool on a Debian 7 system from the Command Line. I will not start any GUI application to build your code. If it is not available in the default package repos that is ﬁne, just include instructions on where I can get it and how to install it. If your build script creates a jar ﬁle, please make sure it is executable. That is, that it can be run with java -jar something.jar ...

**4.3 Test and Auxiliary Files**

If you use or create any test ﬁles or any other auxiliary ﬁles as part of your submission, we would very much like to inspect those as well. Please include them in the submission.

**4.4 README**

Make sure to include a README ﬁle in your project that explains how to build and run your project. Please name this ﬁle README, please make sure that it is in plaint text format, and please place it in the root (top-level) of your project turn in. This ﬁle must explain exactly how to build and run your project.It must also list all libraries that you used, including a URL so that we can look them up. You may include a short writeup in this ﬁle describing your general design motivations. Please mention the location and purpose of any test or auxiliary ﬁles you have included in your submission.

**5 Submitting Your Project**

Please submit your project as a .zip, .tar.gz, .tar.bz, or .tar.xz. Please do not include any compiled code, unless it is an library.

**6 Questions**

If you have any questions or need clariﬁcation about anything please feel free to send an email