University of Reading

Course: BSc. Computer Science
Module: Advanced Databases
Module Convenor: Dr. John Roberts
Student: Vinícius de Oliveira Silva – **Student Number:** 24023627

XML Data Transfer Coursework Report

1. Introduction:

Storing data is a task that has always been a challenge in the computing field. Sometimes, the amount of data to be stored and the unstructured way in which it is organized are challenging matters that computer scientist have to face. By using advanced modelling techniques and some creativity, computing professionals usually are able to abstract the data in a way that makes it easy to store and process, but in some cases, it is not possible to do so.

Some computational problems inherently require their data to be organized in an unstructured way, which makes it difficult to process and store. In this kind of problems, it is common that multiple instances of the same "data class" do not share the same attributes, making it hard to create a model that efficiently abstracts the general case.

In order to solve the problems generated by these situations, it was necessary to create a more flexible way to store data. The eXtensible Markup Language (XML) is a powerful tool proposed by W3C during the 90's that allows its users to store their data in a much more flexible way. By using tags structured as a tree, XML users are able to abstract the information that needs to be stored without worrying about creating a static model that describes the general format of the data.

In this article, two simple applications that make use of this technology are going to be presented. The idea is to show how XML can be used to facilitate the integration between distinct applications that need to access the same data.

In order to develop the applications, the Java programming language was used along with a third-party library called JDOM (Java Document Object Model). The reason why these technologies were chosen is the fact that they are simple to use, easy to understand and efficiently abstract the details of the implementation, allowing the programmer to focus on what is really important, in this case, demonstrating some concepts related to XML handling.

2. Task Description and Implementation Decisions:

Accordingly to the Coursework specification, the objective of this task is to implement a system for recording the details of a reported crime. The system should be designed in the form of two small programs, one for storing the data into an XML file and the other for retrieving the data stored by the first.

The solution provided assumes the existence of a file that contains the input data. It means that the first program does not do any interaction with the user. It basically reads the input data from a file that stores the data in a serialized way and writes it into an XML file, making it possible to both humans and computers to read and understand the data.

The decision of implementing the program this way was made in order to simplify the coding. By abstracting of creating a proper GUI and relying on a pre-existing file containing the input data, time and effort were saved on details and applied on designing an application that properly fulfils its role: demonstrate the use of XML files for storing unstructured data.

The program 2's implementation also does not cover interaction with the user. Its behaviour was designed just to read the XML file produced by program 1 and to generate a text report of the read data in a separated file named "Report.txt".

It is also important to notice that in order to generate the pre-existing file containing the input data, a third program (program 3) was implemented. Its behaviour is extremely simple: It just store some objects into a binary file using Java's native serialization.

3. XML Design

The system was designed in a way that allows it to deal with three different types of crimes: theft, murder and traffic violation. These three crimes were chosen due to the fact that they don't share many attributes, making it to possible to generate unstructured data and therefore, demonstrate how the use of XML can be beneficial in this kind of situations.

In order to describe the crimes, different XML tags were used in each type. The list of tags for each crime is presented below:

- **Murder:** <victim>, <location>, <weapon>, <suspects>, <suspect>;
- **Theft:** <objectStolen>, <objectOwner>, <location>, <suspects>, <suspect>;
- **Traffic Violation:** <driver>, <vehiclePlate>, <offense>, <location>.

Considering the tags presented above, it is possible to create short example of how a typical XML looks like in the context of this system. The example is presented below (**Picture 1**):

```xml
<?xml version="1.0" encoding="UTF-8"?>
<CrimesInfo>
  <crime>
    <type>theft</type>
    <objectStolen>Watch</objectStolen>
    <objectOwner>Ben T.</objectOwner>
    <location>Woodbridge Rd, Greenvale, UK</location>
    <suspects>
      <suspect>Peter Petrelli</suspect>
      <suspect>Nathan Drake</suspect>
    </suspects>
  </crime>
  <crime>
    <type>murder</type>
    <victim>John Snow</victim>
    <weapon>Sword</weapon>
    <location>Winterfell St, Greenvale, UK</location>
    <suspects>
      <suspect>Clark</suspect>
      <suspect>Gordon</suspect>
    </suspects>
  </crime>
  <crime>
    <type>traffic violation</type>
    <driver>Sam W.</driver>
    <vehiclePlate>489 PCE</vehiclePlate>
    <offense>Speeding</offense>
    <location>Greenlake Road, Greenvale, UK</location>
  </crime>
</CrimesInfo>
```

Picture 1

4. Program Design and Execution

The design of both programs was based on Java technology, using the object orientation paradigm. The implementation was developed in a modular way, respecting the encapsulation rules. In order to implement the core of the programs, which is XML handling, a third-party library named “JDOM” was used. It is an open-source library available for free download on jdom.org that provides useful tools for handling XMLs using Java.

As stated before, the execution of the program is based on the data provided by a pre-existing file. This file, named “InputData.dat” was generated by program 3, a short piece of code that simply writes some pre-defined data into the file in a format that is not readable by humans. By comparing this file and the XML generated by program 1, it is possible to see one of the biggest advantages of using XML: the data is stored in a flexible structure that is readable by humans and computers.

The source-code of program 3 is available in the zip file uploaded along with this report, as well as all the other source-codes used to develop the system.

The output of the system as a whole is a text file describing, in plain English, the data stored in the XML file. The output for the XML presented on Picture 1 is represented in **Picture 2.**

```
--------------------------------------------------
Crime: Theft
      Object Stolen: Watch
      Object Owner: Ben T.
      Location: Woodbridge Rd, Greenvale, UK
      Suspects:
            Peter Petrelli
            Natan Drake

--------------------------------------------------

--------------------------------------------------
Crime: Murder
      Victim: John Snow
      Location: Winterfell St, Greenvale, UK
      Weapon: Sword
      Suspects:
            Clark
            Gordon

--------------------------------------------------

--------------------------------------------------
Crime: Traffic Violation
      Driver: Sam W.
      Vehicle Plate: 489 PCE
      Offense: Speeding
      Location: Greenlake Road, Greenvale, UK

--------------------------------------------------
```

Picture 2

5. References:

- *How to create XML file in Java* (2011). Accessed on February 24, 2016, available at http://www.mkyong.com/java/how-to-create-xml-file-in-java-jdom-parser/

- *Java XML – Example JDOM2 Usage*. Accessed on February 24, 2016, available at http://www.studytrails.com/java/xml/jdom2/java-xml-jdom2-example-usage.jsp

- *Connoly, Dan (2003). Development History. Accessed on February 24, 2016, available at https://www.w3.org/XML/hist2002*

- *JDOM Documentation. Accessed on February 24, 2016, available at http://www.jdom.org/docs/apidocs.*