 [1]

Data Structure Project – XML Editor

**Program:**

***Course Code: CSE323***

***Course Name: Data Structure***

***Submitted to:***

***Dr : Islam El-maddah***

**Eng, Amr Ahmed**

**Ain Shams University**

**Faculty of Engineering**

**CSE Department**

**2021**

**Student Personal Information for Group Work**

|  |
| --- |
| **Student Codes:** |

1700267

1700271

1700462

1601583

|  |
| --- |
| **Student Names:** |

Alaa Ibrahim Mohamed Ibrahim Amer

Alaa Shaaban Hussien Ali Shatat

Hussein Mahmoud Fouad El-Sayed

Nahla Mostafa Abdelkareem

Table of Contents

[1. Background 4](#_Toc77190194)

[2. Implementation details 4](#_Toc77190195)

[2.1 The algorithm used in the XML editor 4](#_Toc77190196)

[2.2 Operations 5](#_Toc77190197)

[3. Complexity of operations 5](#_Toc77190198)

[4. References 5](#_Toc77190199)

[5. Working files 5](#_Toc77190200)

# Background

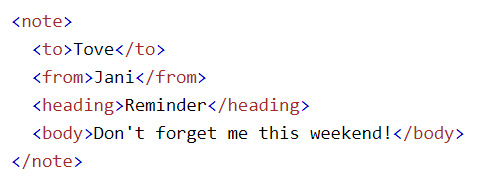
**XML** stands for eXtensible Markup Language, it is similar to HTML language “used in web”, XML was designed to store and transport data. XML is a language that defines a set of rules for encoding a document in a format that is both human-readable and machine-readable. [1]

In **XML** there are:

* Start-tag such as <example>.
* End-tag such as </example>.
* Element content, the characters between the open-tag and the end-tag such as <example>**she was able to program her computer**</example>
* Attributes: An *attribute* is a markup construct consisting of a [name–value pair](https://en.wikipedia.org/wiki/Attribute%E2%80%93value_pair) that exists within a start-tag or empty-element tag such as

<word lex\_id="0">able</word>

**XML** documents may begin with an *XML declaration* that describes some information about themselves. An example is <?xml version="1.0" encoding="UTF-8"?>.

**XML** identifies data using tags, which are identifiers enclosed in angle brackets, each XML document has a single root element, for example: 

# Implementation details

* Language used C#.

## The algorithm used in the XML editor

At first, in the GUI the user should browse and select a XML file and import it, then can choose what operation will be executed, operations are:

* Parse XML file.
* Check and correct errors.
* Format the XML file.
* Convert the XML to Jason.
* Minify the XML file.
* Compress the data.

When the user choose the operation. According to the user choice the function of this operation will be executed and the required result will be printed and the user can store it.

## Operations

OOP is used in order to simplify and organize the code.

Classes used:

* XML class: contains the functions used to execute the user choice. Contains of List of Tags and other variables to handle the operations.
* Tag class: contains a tag name, element content , a list of Tag” child tags”, because each tag in the XML file may contain a child tag maybe we can call it nested tags, so we need to store the parent tag and Childs for it and contains attributes to store the attribute of each tag.
* Tag\_Attributes class: contains the attribute name and the attribute value.

### Parsing Function

### 2.2.2 Correction Function

### 2.2.3 Formatting Function

### 2.2.4 ConvertToJason Function

### 2.2.5 Trim Function

### 2.2.6 Compress Function

# Complexity of operations

# References

|  |  |
| --- | --- |
| [1] | J. Lu, "AN Introduction to XML Query Processing and Keyword Search," in *AN Introduction to XML Query Processing and Keyword Search*. |
|  |  |

# Working files

* Repository link on github:

<https://github.com/AlaaShatat/Data-Structure-project>

* Short video link

# 