

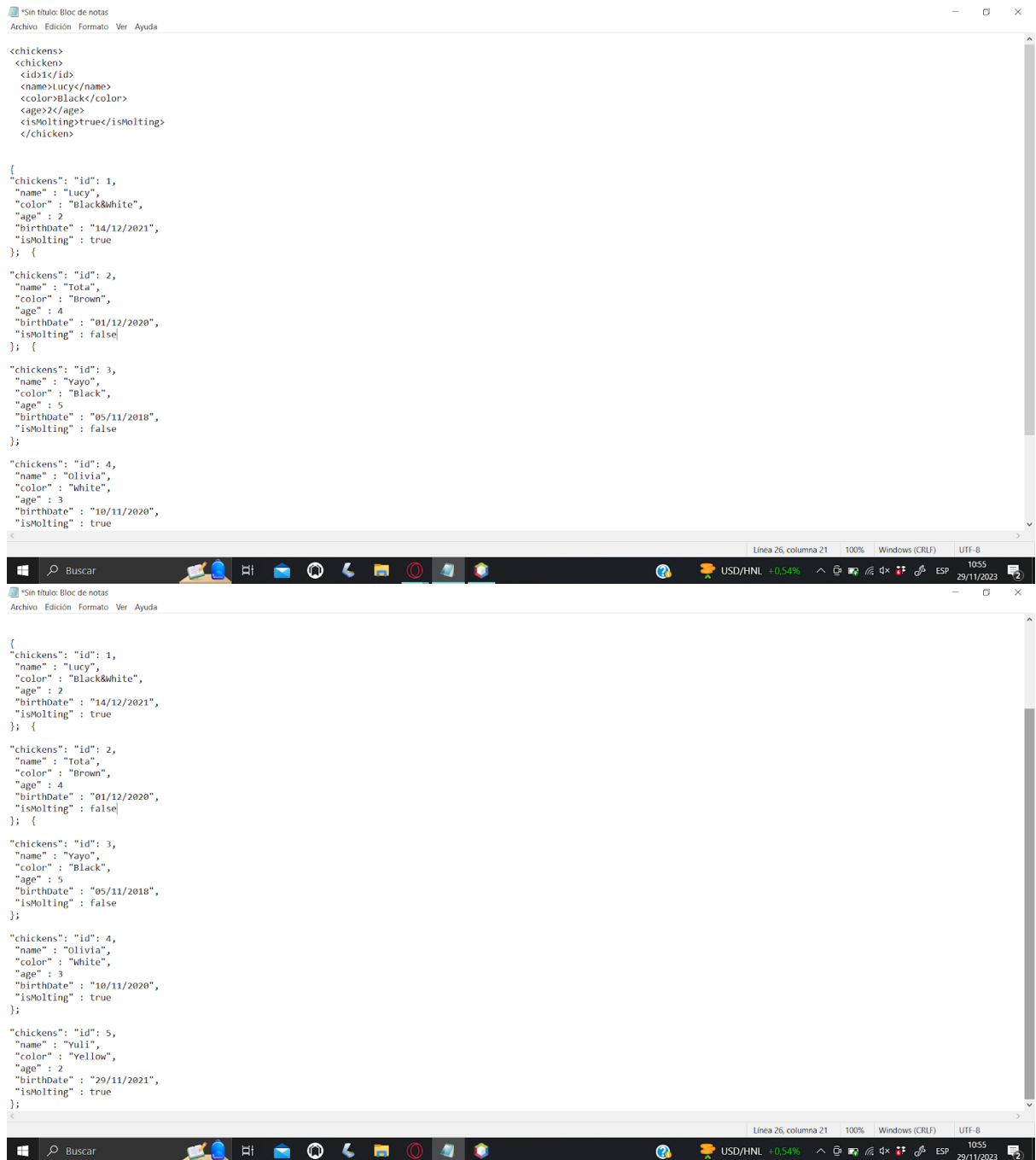
Universidad de las Fuerzas Armadas

CLASS NAME : OBJECT ORIENTED
PROGRAMMING
LASCANO
NRC: 14575
TOPIC:

TEACHER: EDISON

WORKSHOP #: 8

David Gustavo Cepeda Salguero



The image shows a Notepad++ window with the title bar "Sin título: Bloc de notas". The menu bar includes "Archivo", "Edición", "Formato", "Ver", and "Ayuda". The status bar at the bottom indicates "Linea 26, columna 21", "100%", "Windows (CRLF)", and "UTF-8".

The code is divided into two sections. The top section contains XML code for a 'chickens' dataset:

```
<chickens>
  <chicken>
    <id>1</id>
    <name>Lucy</name>
    <color>Black</color>
    <age>2</age>
    <isMolting>true</isMolting>
  </chicken>

  <chicken>
    <id>2</id>
    <name>Tota</name>
    <color>Brown</color>
    <age>4</age>
    <isMolting>false</isMolting>
  </chicken>

  <chicken>
    <id>3</id>
    <name>Yayo</name>
    <color>Black</color>
    <age>5</age>
    <isMolting>false</isMolting>
  </chicken>

  <chicken>
    <id>4</id>
    <name>Olivia</name>
    <color>White</color>
    <age>3</age>
    <isMolting>true</isMolting>
  </chicken>
</chickens>
```

The bottom section contains the corresponding JSON code:

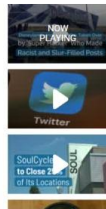
```
{
  "chickens": {
    "id": 1,
    "name": "Lucy",
    "color": "Black&White",
    "age": 2,
    "birthDate": "14/12/2021",
    "isMolting": true
  },
  "chickens": {
    "id": 2,
    "name": "Tota",
    "color": "Brown",
    "age": 4,
    "birthDate": "01/12/2020",
    "isMolting": false
  },
  "chickens": {
    "id": 3,
    "name": "Yayo",
    "color": "Black",
    "age": 5,
    "birthDate": "05/11/2018",
    "isMolting": false
  },
  "chickens": {
    "id": 4,
    "name": "Olivia",
    "color": "White",
    "age": 3,
    "birthDate": "10/11/2020",
    "isMolting": true
  },
  "chickens": {
    "id": 5,
    "name": "Yuli",
    "color": "Yellow",
    "age": 2,
    "birthDate": "29/11/2021",
    "isMolting": true
  }
}
```



[Validate](#) [Documentation](#) [An example](#) [About](#) [Privacy](#)

FEATURED VIDEOS

Powered by [joomla](#)



Disneyland's...
Disneylands Instagram
Was Taken Over , by
Super... [Watch Video](#)

Twitter Sues...
Twitter Sues
Government of India ,
Over Blocked... [Watch Video](#)

SoulCycle to Close...
SoulCycle ,to Close
25% , of Its Locations
CNN reports that... [Watch Video](#)

Amazon...

Validate an XML file

Read [here](#) how to validate your XML files (including referenced DTDs) online with just a few mouse clicks.

An error has been found!

Click on [↖](#) to jump to the error. In the document, you can point at [↖](#) with your mouse to see the error message.

Errors in the XML document:

- 5:21 The reference to entity "White" must end with the ';' delimiter.

XML document:

```
1 <chickens>
2 <chicken>
3 <id>1</id>
4 <name>Lucy</name>
5 <color>Black&White
6 </color>
7 <age>2</age>
8 <isMolting>true</isMolting>
```

The following files have been uploaded so far:

[XML document](#):#

Click on any file name if you want to edit the file.



To format and validate your JSON, just copy + paste it below:

```
1 {
2   "firstName": "John",
3   "lastName": "Doe",
4   "age": 33,
5   "isMarried": true,
6   "children": [
7     "Michael",
8     "Mary",
9     "Mark"
10  ],
11  "pets": [
12    "dog",
13    "cat"
14  ],
15  "address": {
16    "street": "1234 Main St",
17    "city": "New York",
18    "state": "NY",
19    "zip": "10001"
20  },
21  "hobbies": [
22    "reading",
23    "golfing",
24    "fishing"
25  ],
26  "isEmployed": true
27 }
```



Invalid JSON!

Trace: Error: SyntaxError: Unexpected token '}' in JSON at position 100

Expected: { "key": "value", "key": "value", "key": "value" }

Other tools from JSONLint:

- [JSON to XML](#)
- [JSON to CSV](#)

About the JSONLint Editor

JSONLint is a validator and formatter for JSON, a lightweight data interchange format. Copy and paste, directly type, or input a URL in the editor above and let JSONLint tidy and validate your messy JSON code.

What is JSON?

JSON (pronounced as Jasson), stands for "JavaScript Object Notation," is a human-readable and compact solution to represent a complex data structure and facilitate data interchange between systems. It's a widespread data format with a diverse range of applications enabled by its simplicity and semblance to readable text. As such, it's used by most but not all systems for communicating data.

Why Use JSON?

There are several reasons why you should consider using JSON, the key reason being that JSON is independent of your system's programming language, despite being derived from JavaScript. Not only is JSON language-independent, but it also represents data that speaks common elements of many programming languages, effectively making it into a universal data representation understood by all systems.

Other reasons include:

- **Readability** – JSON is human-readable, given proper formatting.
- **Compactness** – JSON data format doesn't use a complex markup structure, unlike XML.
- It's easy to analyse into logical syntactic components, especially in JavaScript.
- Countless JSON libraries are available for most programming languages.

Proper JSON Format

Using JSON doesn't require any JavaScript knowledge, though having such would only improve your understanding of JSON. And though the knowledge of JavaScript isn't necessary, following specific rules is:

- Data is in name/value pairs
- Data is separated by commas
- Objects are encapsulated within the opening and closing curly brackets
- An empty object can be represented by `{ }`
- Arrays are encapsulated within opening and closing square brackets
- An empty array can be represented by `[]`
- A member is represented by a key-value pair, contained in double quotes
- Each member should have a unique key within an object structure
- The value of a member must be contained in double quotes, if it's a string
- Boolean values are represented using the `true` or `false` literals in lower case
- Number values are represented using double-precision floating-point format and shouldn't have leading zeroes
- "Offensive" characters in a string need to be escaped using the backslash character `\`
- Null values are represented by the `null` literal in lower case
- Dates, and similar object types, aren't adequately supported and should be converted to strings
- Each member of an object or array value must be followed by a comma, except for the last one
- The standard extension for the JSON file is `.json`
- The mime type for JSON files is `application/json`

You can achieve proper JSON formatting by following these simple rules. However, if you're unsure about your code, we suggest using this JSONLint Validator and Formatter.

Why Use JSONLint Validator and Formatter?

Programming can be challenging, as it requires enormous attention and excellent knowledge of the programming language, even as simple as JSON. Still, writing code's tricky, and finding an error in JSON code can be a challenging and time-consuming task.

The best way to find and correct errors while simultaneously saving time is to use an online tool such as JSONLint. JSONLint will