Descriptive HTML

Objective

The HyperText Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as <img /> directly introduce content into the page. Other tags such as <p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.
In this lab, you will be introduced to implement a simple grammar for a declarative language that defines the web page instead of the HTML. You will use a tool named JavaCC.

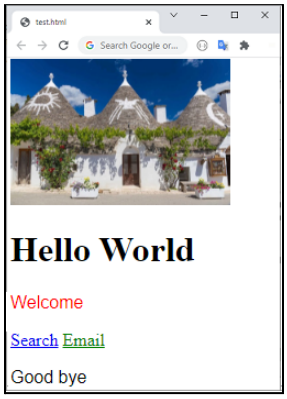
Example

People with no technical background may find learning HTML a difficult thing, so you will make their life easier by designing a language that describes the web page (WHAT the user needs?) instead of telling the browser (HOW to render the page?).
For example, the user will write something like the following

```
ADD IMAGE WITH SOURCE "https://www.w3schools.com/html/pic_trulli.jpg"
ADD HEADING WITH TEXT "Hello World"
ADD PARAGRAPH WITH TEXT "Welcome" AND WITH FONT "Arial" AND WITH COLOR "Red"
ADD LINK WITH TEXT "Search" AND WITH LINK "http://google.com"
ADD LINK WITH TEXT "Email" AND WITH LINK "http://gmail.com" WITH COLOR "Green"
ADD PARAGRAPH WITH WITH FONT "Arial" AND TEXT "Good bye"
```

Which will generate the following HTML

```
<img src="https://www.w3schools.com/html/pic_trulli.jpg" />
<h1>Hello World</h1>
<p style="color:Red;font-family:Arial;">Welcome</p>
<a href="http://google.com">Search</a>
<a href="http://gmail.com" style="color:Green;">Email</a>
<p style="font-family:Arial;">Good bye</p>
```



Grammar

You can start by the following grammar for this descriptive language

```
create →  "ADD" element
element →  img | header | para | url

img → "IMAGE" "WITH" "SOURCE" quote sentence quote
header → "HEADING" decorated_text
para → "PARAGRAPH" decorated_text
url → "LINK" decorated_url

decorated_text → decorated_text "AND" decorated_text | text | color | font

text → "WITH" "TEXT" quote sentence quote
color → "WITH" "COLOR" quote sentence quote
font → "WITH" "FONT" quote sentence quote

sentence → sentence alphanumeric | ε
quote → "\""
alphanumeric → "0"-"9" | "a"-"z" | "A"-"Z" | "/" | ":" | "."
```

Note, the definition for `decorated_url` is left as an exercise to you. It must support the options of `decorated_text` besides defining **"WITH LINK"**

By the end of this lab, you will be able to do two things

1. Validate any given expression and determine if it belongs to this language or not.
2. Evaluating the grammar expressions to generate the equivalent HTML. (JavaCC supports evaluating the grammar expressions.)

Deliverable

- A grammar file with JavaCC JJ format
- A generated parser that parse the language above
- 10 JUnit Tests that verify your Parser is validating different set of expressions
- 5 JUnit Tests that verify your Parser is generating the expected HTML for a different set of expressions

Delivery

- Submission of the assignment using Microsoft teams course page
- Group of 3 students (or 2 students with a permission from the TA)
- Due Date is 18 March

Helping Material

- JavaCC tutorials https://javacc.github.io/javacc/tutorials/
- JUnit tutorial https://www.vogella.com/tutorials/JUnit/article.html

Appendix: JavaCC

Overview

“Java Compiler Compiler (JavaCC) is the most popular parser generator for use with Java applications.
A parser generator is a tool that reads a grammar specification and converts it to a Java program that can recognize matches to the grammar.
In addition to the parser generator itself, JavaCC provides other standard capabilities related to parser generation such as tree building (via a tool called JJTree included with JavaCC), actions and debugging.” **-- source https://javacc.github.io/javacc/**

Setup

1. Download eclipse
2. From menu “Help” → “Eclipse Marketplace …”
3. Search for JavaCC, and select the first plugin shown in the list
4. Install the plugin and restart eclipse
5. From menu “File” → “New” → “Others” then select Java project
6. Create a package named “egtry.hello”
7. Create an empty file named hello.jj
8. Copy the content of this page into that file http://www.egtry.com/tools/javacc/hello
9. Right click the hello.jj file, and from the context menu choose the option of “Compile with javacc | jjtree | jtp”
10. A set of files will be generated that are equivalent to the grammar defined in the hello.jj. This set of files represents a parser for the language defined in the file with extension jj
11. Create a new Java class with a main method with the following code
    ```
    new egtry.hello.Hello(System.in).words();
    ```
12. Run you main method and enter any word followed by numbers (e.g. hello 123)
13. Try out different inputs and see how the parser code will respond to your inputs
14. To get familiar with JavaCC, play with the grammar file hello.jj and regenerate the parser files again (step 9).