OPANGA TEST SOLUTION

Javier Mercado Velázquez

Write a program to parse configuration files made up of white space separated words and sections delimited by curly braces. Here is an example of the configuration file:

runtime {

key1 value1

key2 value2

flag1

system1 {

prop1 value1

prop2 value2

ports 1234 5678 9102

subsystem1 {

prop3 value1 value2 value3

flag2

}

}

more stuff here

}

* Save the parsed configuration in a data structure that allows querying for sections, subsections, key value pairs, and flags.
* Consider how to handle malformed configuration files.
* Write code to test your program, in the form of unit tests appropriate for the programming language of your choice.
* The code should compile and run, as well all the test cases should pass.
* Include in your solution all the instructions necessary to run the program, including how to set up the runtime environment with dependencies and requirements.
* Choose any one of the following programming languages: C, C++, Java, Python, JavaScript.
* Let us know if you have any questions.

gitRepo: <https://github.com/elchangodiaz/Spring/tree/main/OpangaTest>

The solution is developed with java 11.

For this solution the main class call the configFileParser.java class.

This class have the next methos:

validateFile

getSections

getKeys

getFlags

These methods return data structures with the sections of the file.

For example

These are the sections.

{runtime =0, subsystem1 =2, subsystem2 =2, system2 =1, system1 =1}

These are the keys and its values:

[{key1=value1}, {key2=value2}, {prop1=value1}, {prop2=value2}, {ports=1234 5678 9102}, {prop3=value1 value2 value3}, {prop1=value1}, {prop2=value2}, {ports=1234 5678 9102}, {prop3=value1 value2 value3}]

These are the flags

[ flag1, flag2, flag2]

You can add more config sections, keys and flags. And it depends of the validations that are required that the program will check.

The dependencies used are:

Test:

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-api</artifactId>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

Program logic:

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<groupId>org.apache.commons</groupId>

<artifactId>commons-lang3</artifactId>