Activity: Create a Parser for the FOO Language

FOO is a language designed for exchanging data in a human-readable format. FOO's basic types are:

- **Number**
- **String** (double-quoted Unicode, with backslash escaping)
- **Boolean** (true or false)
- **Array** (an ordered sequence of values, comma-separated and enclosed in square brackets; the values do not need to be of the same type)
- **Object** (an unordered collection of key:value pairs with the ':' character separating the key and the value, comma-separated and enclosed in curly braces; the keys must be strings and should be distinct from each other)
- **null** (empty)

Non-significant white space may be added freely around the "structural characters" (i.e. brackets "{ } [ ]", colons ":" and commas ",").

The following example shows the FOO representation of an object that describes a person. The object has string fields for first name and last name, a number field for age, an object representing the person's address and an array of phone number objects.

```
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 25,
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": 10021
    },
    "phoneNumbers": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "fax",
            "number": "646 555-4567"
        }
    ]
}
```

Complete the following ANTLR grammar for FOO:

```
grammar FooLang;


foo : /* You, complete me. */  ;


/*
 * Tokens
 */

STRING :  '"' (ESC | ~["\\])* '"' ;

fragment ESC :   '\\' (["\\/bfnrt] | UNICODE) ;
fragment UNICODE : 'u' HEX HEX HEX HEX ;
fragment HEX : [0-9a-fA-F] ;

NUMBER
    :   '-'? INT '.' INT EXP?   // 1.35, 1.35E-9, 0.3, -4.5
    |   '-'? INT EXP            // 1e10 -3e4
    |   '-'? INT                // -3, 45
    ;
fragment INT :   '0' | [1-9] [0-9]* ; // no leading zeros
fragment EXP :   [Ee] [+\-]? INT ; // \- since - means
"range" inside [...]

WS  :   [ \t\n\r]+ -> skip ;
```