```
Name: Preston Tighe
Class: CSE 3342 - Programming Languages
Date: 11-28-2016
Assignment: Assignment 12B

GitHub: https://github.com/CSE3342/Assignment12

URL: http://lyle.smu.edu/~prestont/3342/dataService.php

Tracking: people

Contents: MyLanguage.g4, MyLanguageRunner.java, dataService.php, output.pdf
```

File - dataService.php

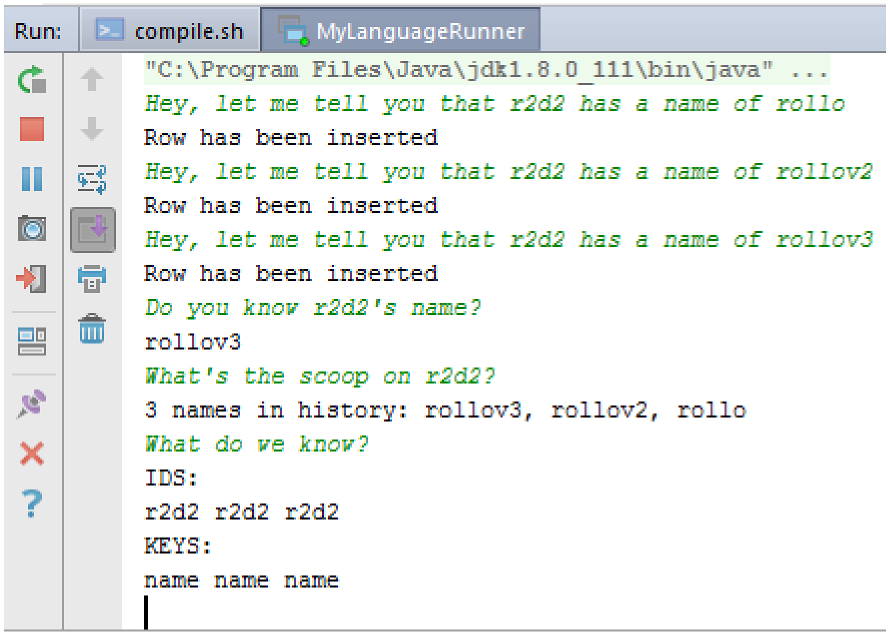
```php
<?php
//Preston Tighe
//Programming Languages
//10-19-16

class Database {
    private $database_name = 'database.json';
    private $id;
    private $key;
    private $val;
    private $rows = array();

    public function __construct () {
        //Check valid GET params
        $this->_check_params();

        //Store GET params
        $this->_store_params();

        //Check valid ID if provided
        $this->_check_id();

        //Clear database
        if(!empty($_GET['clear'])){
            $this->_clear_database();
            exit;
        }

        //Fetch previous database
        $this->_fetch_database();

        //Return value where id & key
        if(!empty($this->id) && empty($this->key) && empty($this->val)){
            echo $this->_lookup_id();
            exit;
        }

        //Return value where id & key
        if(!empty($this->id) && !empty($this->key) && empty($this->val)){
            echo $this->_lookup_key();
            exit;
        }

        //Print all rows
        if(empty($this->id) && empty($this->key) && empty($this->val)){
            echo $this->_print_database();
            exit;
        }

        //Store row
        if(!empty($this->id) && !empty($this->key) && !empty($this->val)){
            $this->_insert_row();
            exit;
        }

        //Error handling
        if(empty($this->id) && !empty($this->key) && empty($this->val)){
            throw new Exception('ID is required.');
        }
        if(empty($this->id) && empty($this->key) && !empty($this->val)){
            throw new Exception('ID & key is required.');
        }
    }
    private function _check_params(){
        //Check valid GET params
        $valid_keys = array('id','key','val','clear');
        $invalid_keys = array();
        foreach($_GET as $key => $value){
            if(!in_array($key,$valid_keys)){
                $invalid_keys[] = $key;
            }
        }
        if(!empty($invalid_keys)){
            throw new Exception('Invalid keys: ' .
                implode(', ',$invalid_keys) .
                ', expecting keys: ' . implode(', ',$valid_keys));
        }
    }
    private function _store_params(){
        $this->id = !empty($_GET['id']) ? $_GET['id'] : null;
        $this->key = !empty($_GET['key']) ? $_GET['key'] : null;
        $this->val = !empty($_GET['val']) ? $_GET['val'] : null;
    }
    private function _check_id(){
        if(!empty($this->id) && !preg_match('/[a-zA-Z][\w]*/',$this->id)){
            throw new Exception('The ID can only be the letters a-z or A-Z.');
        }
    }
    private function _clear_database(){
        $this->rows = array();
        $this->_save_database();
    }
    private function _fetch_database(){
        if(file_exists($this->database_name)){
            $this->rows = json_decode(
                file_get_contents($this->database_name), true);
        }
    }
    private function _lookup_id(){
        $return_data = array();
        foreach($this->rows as $row){
            if($row['id'] == $this->id){
                $return_data[] = $row;
            }
        }
        if(empty($return_data)){
            throw new Exception('Could not find any rows with ID #' .
                $this->id . '.');
        }

        $names = array();
        foreach($return_data as $item){
            $names[] = $item['val'];
        }
        return count($return_data) . ' names in history: ' .
            implode(', ', array_reverse($names));
    }
    private function _lookup_key(){
        foreach(array_reverse($this->rows) as $row){
            if($row['id'] == $this->id){
                if($row['key'] == $this->key){
                    return $row['val'];
                }
            }
        }
        throw new Exception('Could not find a row with ID #' . $this->id .
            ' and key `' . $this->key . '`.');
    }
    private function _print_database(){
        $message = '';
        $ids = array();
        $keys = array();
        foreach($this->rows as $row){
            $ids[] = $row['id'];
            $keys[] = $row['key'];
```

File - dataService.php
```php
        }
        $message .= 'IDS:' . "\n";
        $message .= implode(' ', $ids);
        $message .= "\n" . 'KEYS:' . "\n";
        $message .= implode(' ', $keys);
        return $message;
    }
    private function _insert_row(){
        $this->rows[] = array(
            'id' => $this->id,
            'key' => $this->key,
            'val' => $this->val
        );
        $this->_save_database();
        echo 'Row has been inserted';
    }
    private function _save_database(){
        file_put_contents($this->database_name, json_encode($this->rows));
    }
}

try{
    $database = new Database();
}catch(Exception $e){
    echo 'Error: ' . $e->getMessage();
}

```

```java
import org.antlr.v4.runtime.ANTLRInputStream;
import org.antlr.v4.runtime.CommonTokenStream;

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.InputStream;
import java.io.InputStreamReader;

public class MyLanguageRunner {

    public static void main(String[] args) throws Exception {

        // check if we want to use a file as input or System.in
        String inputFile = null;
        if (args.length > 0) inputFile = args[0];
        InputStream is = System.in;
        if (inputFile != null) {
            is = new FileInputStream(inputFile);
        }

        BufferedReader br = new BufferedReader(new InputStreamReader(is));
        String expr = br.readLine(); // get first line of input
        int line = 1; // track line numbers




        // create a Parser that we will reuse for each line of input
        // ** change name of Parser to match your Parser name

        // we will share this single parser instance with different lexers

        MyLanguageParser parser = new MyLanguageParser(null);
        parser.setBuildParseTree(false); // don't need trees

        // as long as we keep getting input we create a new LEXER that will
        // generate a new set of TOKENS to feed to our parser.

        while (expr != null) { // while we have more lines of input
            // create new lexer and token stream for each line (expression)
            ANTLRInputStream input = new ANTLRInputStream(expr + "\n");

            // ** change name of Lexer to match your Lexer
            MyLanguageLexer lexer = new MyLanguageLexer(input);

            // do some lexer work
            lexer.setLine(line); // notify lexer of input position
            lexer.setCharPositionInLine(0);
            CommonTokenStream tokens = new CommonTokenStream(lexer);

            // pass our TOKENS to the parser
            parser.setInputStream(tokens); // notify parser of new token stream

            // ** change 's' to your starting parser rule
            parser.root_rule(); // start the parser to match rule s

            expr = br.readLine(); // see if there's another line
            line++;
        }
    }
}
```

```
grammar MyLanguage;

@header {
    import java.io.*;
    import java.net.URL;
    import java.nio.charset.Charset;
}

@parser::members {
    final private static String public_url = "http://lyle.smu.edu/~prestont/" +
        "3342/dataService.php";
    private static String setIdKeyVal(String id, String key, String val) {
        return readFromUrl(public_url + "?id=" + id + "&key=" + key +
                "&val=" + val);
    }
    private static String getIdKey(String id, String key) {
        return readFromUrl(public_url + "?id=" + id + "&key=" + key);
    }
    private static String getId(String id) {
        return readFromUrl(public_url + "?id=" + id);
    }
    private static String getDatabase() {
        return readFromUrl(public_url);
    }
    private static String readFromUrl(String url) {
        try {
            InputStream is = new URL(url).openStream();
            BufferedReader rd = new BufferedReader(
                    new InputStreamReader(is, Charset.forName("UTF-8")));
            StringBuilder sb = new StringBuilder();
            int cp;
            while ((cp = rd.read()) != -1) {
                sb.append((char) cp);
            }
            return sb.toString();
        }catch(Exception e) {
            System.out.println("Exception thrown  :" + e);
            return "";
        }
    }
}

root_rule : (set_id_key_val | get_id_key | get_id | get_database) ;
set_id_key_val : 'Hey, let me tell you that ' id=ITEM ' has a ' key=ITEM
    ' of ' val=ITEM { System.out.println(
                            setIdKeyVal($id.text,$key.text,$val.text)
                          );
                        } ;
get_id_key : 'Do you know ' id=ITEM '\'s ' key=ITEM '?' {
    System.out.println(getIdKey($id.text,$key.text)); } ;
get_id : 'What\'s the scoop on ' id=ITEM '?' {
    System.out.println(getId($id.text)); } ;
get_database : 'What do we know?' { System.out.println(getDatabase()); } ;

ITEM : [a-zA-Z][a-zA-Z_$0-9]+ ;
WS : [ \t\r\n]+ -> skip ;
```

Output



```
"C:\Program Files\Java\jdk1.8.0_111\bin\java" ...
Hey, let me tell you that r2d2 has a name of rollo
Row has been inserted
Hey, let me tell you that r2d2 has a name of rollov2
Row has been inserted
Hey, let me tell you that r2d2 has a name of rollov3
Row has been inserted
Do you know r2d2's name?
rollov3
What's the scoop on r2d2?
3 names in history: rollov3, rollov2, rollo
What do we know?
IDS:
r2d2 r2d2 r2d2
KEYS:
name name name
```