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)){
```

```
46              echo $this->_print_database();
47              exit;
48          }
49
50          //Store row
51          if(!empty($this->id) && !empty($this->key) && !empty($this->val)){
52              $this->_insert_row();
53              exit;
54          }
55
56          //Error handling
57          if(empty($this->id) && !empty($this->key) && empty($this->val)){
58              throw new Exception('ID is required.');
59          }
60          if(empty($this->id) && empty($this->key) && !empty($this->val)){
61              throw new Exception('ID & key is required.');
62          }
63      }
64      private function _check_params(){
65          //Check valid GET params
66          $valid_keys = array('id','key','val','clear');
67          $invalid_keys = array();
68          foreach($_GET as $key => $value){
69              if(!in_array($key,$valid_keys)){
70                  $invalid_keys[] = $key;
71              }
72          }
73          if(!empty($invalid_keys)){
74              throw new Exception('Invalid keys: ' .
75                  implode(', ',$invalid_keys) .
76                  ', expecting keys: ' . implode(', ',$valid_keys));
77          }
78      }
79      private function _store_params(){
80          $this->id = !empty($_GET['id']) ? $_GET['id'] : null;
81          $this->key = !empty($_GET['key']) ? $_GET['key'] : null;
82          $this->val = !empty($_GET['val']) ? $_GET['val'] : null;
83      }
84      private function _check_id(){
85          if(!empty($this->id) && !preg_match('/[a-zA-Z][\w]*/',$this->id)){
86              throw new Exception('The ID can only be the letters a-z or A-Z.');
87          }
88      }
89      private function _clear_database(){
90          $this->rows = array();
```

```
91          $this->_save_database();
92      }
93      private function _fetch_database(){
94          if(file_exists($this->database_name)){
95              $this->rows = json_decode(
96                  file_get_contents($this->database_name), true);
97          }
98      }
99      private function _lookup_id(){
100         $return_data = array();
101         foreach($this->rows as $row){
102             if($row['id'] == $this->id){
103                 $return_data[] = $row;
104             }
105         }
106         if(empty($return_data)){
107             throw new Exception('Could not find any rows with ID #' .
108                 $this->id . '.');
109         }
110
111         $names = array();
112         foreach($return_data as $item){
113             $names[] = $item['val'];
114         }
115         return count($return_data) . ' names in history: ' .
116             implode(', ', array_reverse($names));
117     }
118     private function _lookup_key(){
119         foreach(array_reverse($this->rows) as $row){
120             if($row['id'] == $this->id){
121                 if($row['key'] == $this->key){
122                     return $row['val'];
123                 }
124             }
125         }
126         throw new Exception('Could not find a row with ID #' . $this->id .
127             ' and key `' . $this->key . '`.');
128     }
129     private function _print_database(){
130         $message = '';
131         $ids = array();
132         $keys = array();
133         foreach($this->rows as $row){
134             $ids[] = $row['id'];
135             $keys[] = $row['key'];
```

File - dataService.php

```
136         }
137         $message .= 'IDS:' . "\n";
138         $message .= implode(' ', $ids);
139         $message .= "\n" . 'KEYS:' . "\n";
140         $message .= implode(' ', $keys);
141         return $message;
142     }
143     private function _insert_row(){
144         $this->rows[] = array(
145             'id' => $this->id,
146             'key' => $this->key,
147             'val' => $this->val
148         );
149         $this->_save_database();
150         echo 'Row has been inserted';
151     }
152     private function _save_database(){
153         file_put_contents($this->database_name, json_encode($this->rows));
154     }
155 }
156 
157 try{
158     $database = new Database();
159 }catch(Exception $e){
160     echo 'Error: ' . $e->getMessage();
161 }
162 
```