CSCI2134
Assignment 2

Instructor: Chris Whidden

Due: 11:59 PM, Friday, October 16, 2020

Background: JSON

JSON, the (JavaScript Object Notation) is a widely-used encoding format that makes it easy to exchange data. JSON is a text-based (human readable) format, which makes it ideal for many applications as well as easy to debug. For this assignment, you will need to research the JSON format on the web. For example: `https://www.json.org`. A brief summary is provided here, but you are encouraged to do more in-depth research.

JSON encodes data using objects, arrays, and primitive values. A JSON object is encoded as zero or more key-value pairs, where the key is a string and a value can be an object, an array, a string, a number, `true`, `false`, or `null`. A *string* is any piece of text enclosed in double quotes, e.g., `"Hello World"` and a number is either an integer or a decimal value, e.g., `42` and `3.14`. Objects begin with an left brace ({), contain zero or more key-value pairs, and end with a right brace (}), e.g.,

```
{
  "subject code" : "CSCI",
  "course number"  : 2110,
  "pre-reqs" : [
    "CSCI 1110",
    "CSCI 1101"
  ],
  "exclusions" : null,
  "core" : true,
  "textbook" : {
    "author" : "S. Venugopal",
    "title" : "Data Structures Outside In with Java",
    "publisher" : Prentice Hall",
    "year" : 2007,
    "ISBN-10" : "0-13-198619-8"
  }
}
```

In an object, key-value pairs are denoted *key : value* and multiple key-value pairs are separated by commas (`,`). In JSON, an array is a sequence of values of any kind. An array begins with a left square bracket (`[`), has zero or more values, separated by commas, and ends with a right square bracket (`]`). In the example above the array contains two strings.

If you are not not clear on the structure of JSON, your first task is to review a JSON tutorial, such as the one here: `https://www.tutorialspoint.com/json/index.htm`.

Problem: JSON Object Equality

Determining if two JSON objects are equal is a common and useful operation. Two JSON objects are equal if they have the same set of key-value pairs (not necessarily in the same order), where for each key, the values are equal. In the case of primitive values such as strings, numbers, `true`, `false`, and `null`, two values are equal if they are the same. In the case of arrays, two arrays are equal if both arrays have the same number of elements and they are pairwise equal. I.e., for each array element, the values stored in both arrays are equal For example, the following two figures are examples of equal and unequal objects.

```
{
  "subject code" : "CSCI",
  "course number"  : 2110,
  "pre-reqs" : [
    "CSCI 1110",
    "CSCI 1101"
  ],
  "exclusions" : null,
  "core" : true
}
```

```
{
  "core" : true,
  "course number"  : 2110,
  "exclusions" : null,
  "pre-reqs" : [
    "CSCI 1110",
    "CSCI 1101"
  ],
  "subject code" : "CSCI"
}
```

Figure 1: These JSON objects are equal.

```
{
  "subject code" : "CSCI",
  "course number"  : 2110,
  "pre-reqs" : [
    "CSCI 1110",
    "CSCI 1101"
  ],
  "exclusions" : null,
  "core" : true
}
```

```
{
  "subject code" : "CSCI",
  "course number"  : 2110,
  "pre-reqs" : [
    "CSCI 1101"
    "CSCI 1110",
  ],
  "exclusions" : null,
  "core" : true
}
```

Figure 2: These JSON objects are not equal, because the arrays are not equal.

Your task is to create a program to determine if two JSON objects are equal.

Write a program called `JSim.java` that reads in two JSON objects from the console and determines if they are the equal. Your `JSim.java` program must contain the `main()` method where your program starts running.

Input

The input consists of two JSON objects, one after the other. The input is via the console (`System.in`). You may assume that there will be no error in the input.

Reading Input with *JSONScanner*

The provided *JSONScanner* class can be used to read in the JSON objects from the console. This class has a `next()` method, just like the standard *Scanner* class. But, instead of returning the next word in the input, the `next()` method of the *JSONScanner* class returns the next token in the input. A token is one of

token	description or example	token	description or example
{	left brace	*string*	`"Hello World"`
}	right brace	*number*	`42` or `3.14`
`[`	left square bracket	`true`	
`]`	right square bracket	`false`	
:	colon	`null`	
,	comma		

The *JSONScanner* class is easy to use. First, instantiate an object from the class, passing in `System.in`. Then, repeatedly call `next()` on this object to get one token after the other (see example below). The `hasNext()` method is used to check if another token is available.

```
public void main(String [] args) {
  JSONScanner scanner = new JSDNScanner(System.in);
  ...
    String token = scanner.next();
  ...
```

Processing

The input will contain exactly two JSON objects, one after the other.

Two objects are equal if they have the same set of key-value pairs, which may be in different orders. Two arrays are equal if they are of the same size and are pair-wise equal. Otherwise, two values are equal if they are the same.

An object is said to be in *sorted* form if the key-value pairs are in sorted (alphabetical) order and all objects inside the object are also in *sorted* form. For example, in Figure 1, the object on the left is not in *sorted* form, while the object on the right is in *sorted* form.

Output

If the two objects are equal your program should output one of the objects in *sorted* form (formatted as described below), followed by

```
The objects are equal.
```

If the two objects are not equal your program should output both objects in *sorted* form (formatted as described below), followed by

```
The objects are not equal.
```

Object Output Format

When outputting an object, the left and right braces should be on separate lines, with the key-value pairs indented two spaces. There should be single spaces between the key, the colon, and the value. If the value is an object or an array, the left brace or bracket should be on the same line. All the key-value pairs or values should be indented two more spaces. The right brace or bracket should be on separate line, with the same indentation as the key-value pair. A comma separating key-value pairs in an object or values in an array should immediately follow the key-value pair or value. Please see Figures 1 and 2 for examples.

Examples

Sample Input

```
{
  "subject code" : "CSCI",
  "course number"  : 1110,
  "pre-reqs" : [
  ],
  "core" : true
}
{
  "subject code" : "CSCI",
  "course number"  : 1110,
  "core" : true,
  "pre-reqs" : [
  ]
}
```

Sample Output

```
{
  "core" : true,
  "course number"  : 1110,
  "pre-reqs" : [
  ],
  "subject code" : "CSCI"
}
The objects are equal.
```

Sample Input

```
{
  "subject code" : "CSCI",
  "course number"  : 1110,
  "pre-requisies" : [
  ],
  "exclusions" : null,
  "core" : true
}
{
  "subject code" : "CSCI",
  "course number"  : 1110,
  "core" : true,
  "pre-reqs" : [
  ],
  "exclusions" : null
}
```

Sample Output

```
{
  "core" : true,
  "course number"  : 1110,
  "exclusions" : null,
  "pre-requisies" : [
  ],
  "subject code" : "CSCI"
}
{
  "core" : true,
  "course number"  : 1110,
  "exclusions" : null,
  "pre-reqs" : [
  ],
  "subject code" : "CSCI"
}
The objects are not equal.
```

Hints and Suggestions

- You will either need to use recursion or a stack (or both) to solve this problem.
- Java has both stacks (`java.utils.Stack`) and lists (`java.utils.ArrayList`) built in. As well as binary search (`java.utils.Arrays`).
- The sample solution uses a 2-pass algorithm. The first pass reads in the two JSON objects and stores them. This can be done either iteratively with a stack or recursively. The second pass determines if the two objects are the same. This is easier to do recursively. The third pass prints out the first or both objects, and is also easier to do recursively. Note: The solution cannot be done in a single pass.
- There is not a lot of code to write. The sample solution is under 170 lines of code.
- Your code **must** compile. If it does not compile, you will receive a 0 on the assignment.
- Your code must be well commented and indented. Please see the Code Style Guidelines for this course on Brightspace.
- You may assume that all input will be correct.