**JSON**

JSON or JavaScript Object Notation is a lightweight text-based open standard designed for human-readable data interchange. Conventions used by JSON are known to programmers, which include C, C++, Java, Python, Perl, etc.

* JSON stands for JavaScript Object Notation.
* The format was specified by Douglas Crockford.
* It was designed for human-readable data interchange.
* It has been extended from the JavaScript scripting language.
* The filename extension is **.json**.
* JSON Internet Media type is **application/json**.
* The Uniform Type Identifier is public.json.

## **Uses of JSON**

* It is used while writing JavaScript based applications that includes browser extensions and websites.
* JSON format is used for serializing and transmitting structured data over network connection.
* It is primarily used to transmit data between a server and web applications.
* Web services and APIs use JSON format to provide public data.
* It can be used with modern programming languages.

## **Characteristics of JSON**

* JSON is easy to read and write.
* It is a lightweight text-based interchange format.
* JSON is language independent.

## **Simple Example in JSON**

{

"book": [

{

"id":"01",

"language": "Java",

"edition": "third",

"author": "Herbert Schildt"

},

{

"id":"07",

"language": "C++",

"edition": "second",

"author": "E.Balagurusamy"

}

]

}

## **Environment**

Before you start with encoding and decoding JSON using Java, you need to install any of the JSON modules available. we have to download and install [JSON.simple](https://code.google.com/p/json-simple/) and have to add the location of **json-simple-1.1.1.jar** file to the environment variable CLASSPATH.

## **Mapping between JSON and Java entities**

JSON.simple maps entities from the left side to the right side while decoding or parsing, and maps entities from the right to the left while encoding.

|  |  |
| --- | --- |
| **JSON** | **Java** |
| string | java.lang.String |
| number | java.lang.Number |
| true|false | java.lang.Boolean |
| null | null |
| array | java.util.List |
| object | java.util.Map |

On decoding, the default concrete class of *java.util.List* is *org.json.simple.JSONArray* and the default concrete class of *java.util.Map* is *org.json.simple.JSONObject*.

## **Encoding JSON in Java**

Following is a simple example to encode a JSON object using Java JSONObject which is a subclass of java.util.HashMap. No ordering is provided. If you need the strict ordering of elements, use JSONValue.toJSONString ( map ) method with ordered map implementation such as java.util.LinkedHashMap.

import org.json.simple.JSONObject;

class JsonEncodeDemo {

public static void main(String[] args){

JSONObject obj = new JSONObject();

obj.put("name", "foo");

obj.put("num", new Integer(100));

obj.put("balance", new Double(1000.21));

obj.put("is\_vip", new Boolean(true));

System.out.print(obj);

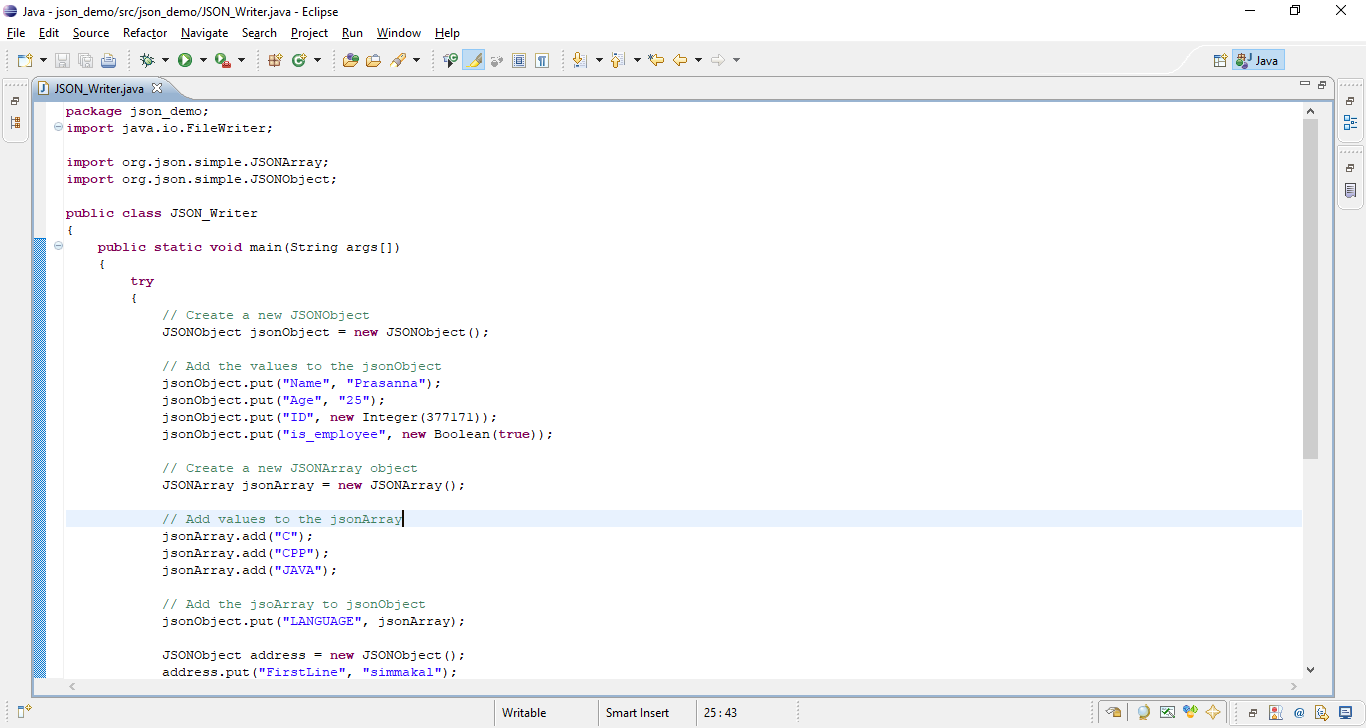
}

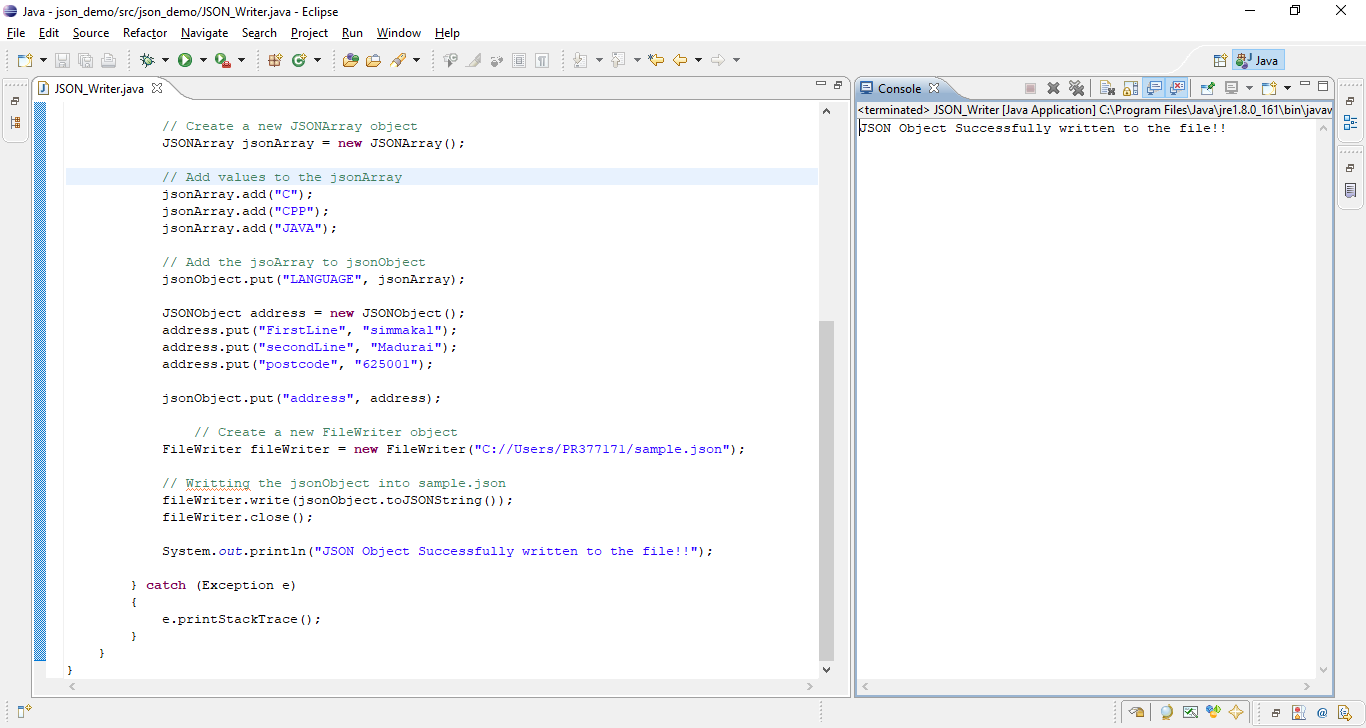
}

On compiling and executing the above program the following result will be generated −

{"balance": 1000.21, "num":100, "is\_vip":true, "name":"foo"}

**Demo Program:**





* Create a new object for **JSONObject**, using the **put()** method of the **jsonObject** the add they **key** and **value** pairs into it.

// Create a new JSONObject

JSONObject jsonObject = **new** JSONObject();

// Add the values to the jsonObject

jsonObject.put("Name", "Prasanna");

jsonObject.put("Age", "25");

jsonObject.put("ID", **new**Integer(377171));

jsonObject.put("is\_employee", **new** Boolean(**true**));

* Create a new object for **JSONArray** to add the list of **countries**, using the **add()** method add the countries into it.

// Create a new JSONArray object

JSONArray jsonArray = **new** JSONArray();

// Add values to the jsonArray

jsonArray.add("C");

jsonArray.add("CPP");

jsonArray.add("JAVA");

* Now, add the **jsonArray** into **jsonObject**

// Add the jsoArray to jsonObject

jsonObject.put("LANGUAGE", jsonArray);

* Finally, create a new object for **FileWriter** and using the **write()** method write the **jsonObject** into the file.

// Writting the jsonObject into sample.json

fileWriter.write(jsonObject.toJSONString());

When we open the **sample.json** file, we will be having the **JSON** written in it.

