Documentação complementar
- JSON - https://www.tutorialspoint.com/json/json_overview.htm
- JSON - https://www.json.org/
- GSON - https://github.com/google/gson/blob/master/UserGuide.md
- GSON - https://www.tutorialspoint.com/gson/gson_tree_model.htm

O presente documento tem por objetivo apresentar uma introdução básica à notação json e à biblioteca gson, proposta na resolução do trabalho prático de recurso. No cabeçalho, podem ser encontrados, recursos auxiliares para esclarecimento de dúvidas relativos a estes tópicos.

Json

Json (JavaScript Object Notation) é um formato de texto que guarda os dados num formato próprio definido como um standard para trocar dados de fácil leitura.
Um objeto em *json* tem o formato definido na figura 1. Dentro das chavetas encontram-se um conjunto de pares chave – valor, em que chave é o nome da propriedade do objeto e o valor a representação do valor de dados.

```
{
        "chave1": "valor1",
        "chave2": "valor2"
}
```

Figura 1 - Objeto Json

Podemos agrupar vários objetos *json* podemos usar *array* tal como demonstrado na figura 2. Dentro dos parêntesis retos podemos agrupar um conjunto de objetos *json* separados por uma vírgula.

```
[
    {
        "chave1": "valor1",
        "chave2": "valor2"

    },
    {
        "chave1": "valor3",
        "chave2": "valor4"

    }
]
```

Figura 2 - Objeto Json´

Podemos também agrupar *arrays* e objetos de forma hierárquica como apresentado nas figuras 3. Os exemplos fornecidos não cobrem todas as combinações possíveis no formato *json*. Servem apenas para demonstrar a versatilidade do formato de dados. Mais exemplos do formato *json* podem ser encontrados na documentação oficial sobre *json* e nos tutoriais fornecidos.

```json
{
        "chave1": [1,2,3],
        "chave2": "valor2"

    },
    {
        "chave1": "valor3",
        "chave3": {
            "subchave1":"subvalor1"
        }

    }
```

```json
[
    {
        "chave1": "valor3",
        "chave3": [
            {
                "subchave1":"subvalor1"
            },
            {
                "subchave1":"subvalor2"
            }
        ]
    }
]
```

Figura 3 - Exemplos de ficheiros json com arrays e objetos

Gson

O Google Gson é uma biblioteca baseada em Java *open source* desenvolvida pela Google. Um dos objetivos desta ferramenta é facilitar a serialização de dados em objetos Java para JSON e vice-versa. No enunciado do trabalho prático é proposto a sua utilização no trabalho prático, mas podem também usar bibliotecas alternativas para o mesmo efeito.
Tendo por base o um ficheiro de dados em *json* no formato da figura 4, vamos traduzir a informação presente num *array* de objetos com a propriedade *nome* e *idade* em objetos java.

```json
[
    {
        "nome": "Pedro",
        "idade": "29"

    },
    {
        "nome": "Luís",
        "idade": "13"

    }
]
```

Figura 4 – Ficheiro exemplo.json

Primeiro, podemos devemos definir uma estrutura de dados adequado para guardar a informação em objetos java, por exemplo a classe Pessoa na figura 5. De seguida podemos usar o código presente na figura 6 para ler a informação do ficheiro *json* para objetos da classe Pessoa em java.

```java
public class Pessoa {

    private String nome;
    private int idade;

    public Pessoa(String nome, int idade) {
        this.nome = nome;
        this.idade = idade;
    }

}
```

Figura 5 – Ficheiro Pessoa.java

```java
public static void main(String[] args) {
    Pessoa pessoas[] = new Pessoa[2];
    String path = "exemplo.json";
    BufferedReader reader = null;
    try {
        reader = new BufferedReader(new FileReader(path));
        //Carregamos o buffer reader para um objeto JsonStreamParser da
        //biblioteca gson
        JsonStreamParser p = new JsonStreamParser(reader);
        //O ficheiro começa começa por um array por isso usamos o método
        //next() e fazemos casting para um objecto do tipo JsonArray
        JsonArray arr = (JsonArray) p.next();
        int arr_size = arr.size();
        //Vamos iterar sobre todos os objetos presentes neste JsonArray
        for (int i = 0; i < arr_size; i++) {
            //Retiramos um elemento do JsonArray e faremos casting para
            //um JsonObject
            JsonElement arrayElement = arr.get(i);
            JsonObject obj = arrayElement.getAsJsonObject();

            //Retiramos as propriedades de um JsonObject
            if (obj.has("nome") && obj.has("idade")) {
                //Retiramos as propriedades de um JsonObject
                String temp_nome = obj.get("nome").getAsString();
                int temp_idade = obj.get("idade").getAsInt();
                //Criamos um novo objeto pessoa no array de pessoas
                pessoas[i] = new Pessoa(temp_nome, temp_idade);
            }
        }
    } catch (FileNotFoundException ex) {
        ex.printStackTrace();
    } finally {
        try {
            reader.close();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

Figura 6 - Método main para carregar o pessoas no ficheiro exemplo.json para um array de Pessoas

O exemplo apresentado é apenas o mínimo exemplo funcional e tem o intuito apenas de servir de ponto de partida para o uso da biblioteca *gson*. Existem outras formas alternativas para o mesmo efeito que não estão cobertas nesta demonstração e que podem usar também. Devem utilizar os recursos sobre *gson* disponibilizados como ajuda para a resolução do trabalho prático.