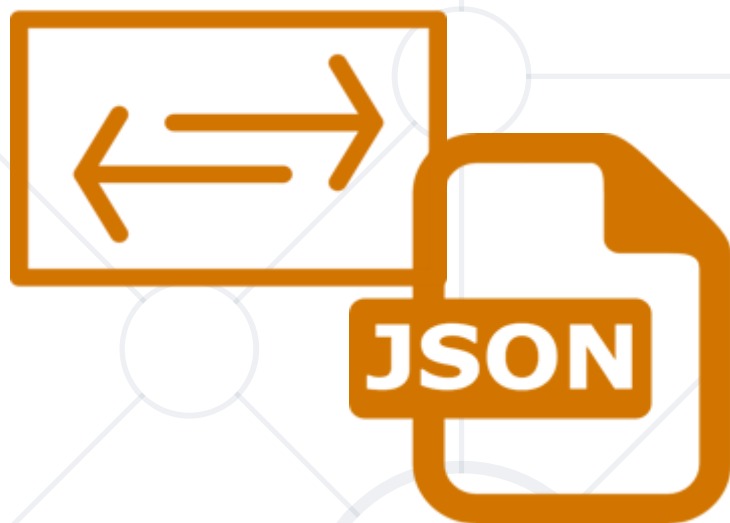


JSON

Exporting and Importing Data from JSON Format



SoftUni Team

Technical Trainers



SoftUni



Software University

<https://softuni.bg>

Table of Contents

1. JSON.

2. GSON.





sli.do
#Java-DB



JSON

Transmitting Data Objects Via Attribute-value Pairs

- JavaScript Object Notation
 - Human-readable format to transmit **data objects** consisting of **attribute–value pairs** and **arrays**
 - Subset of JavaScript syntax
- Supports several data types:
 - Number, String, Boolean, Array, Object, null

JSON Example

person.json

```
{  
  "firstName": "Daniel",  
  "lastName": "Sempere",  
  "age": 24,  
  "isMarried": true  
}
```

Key

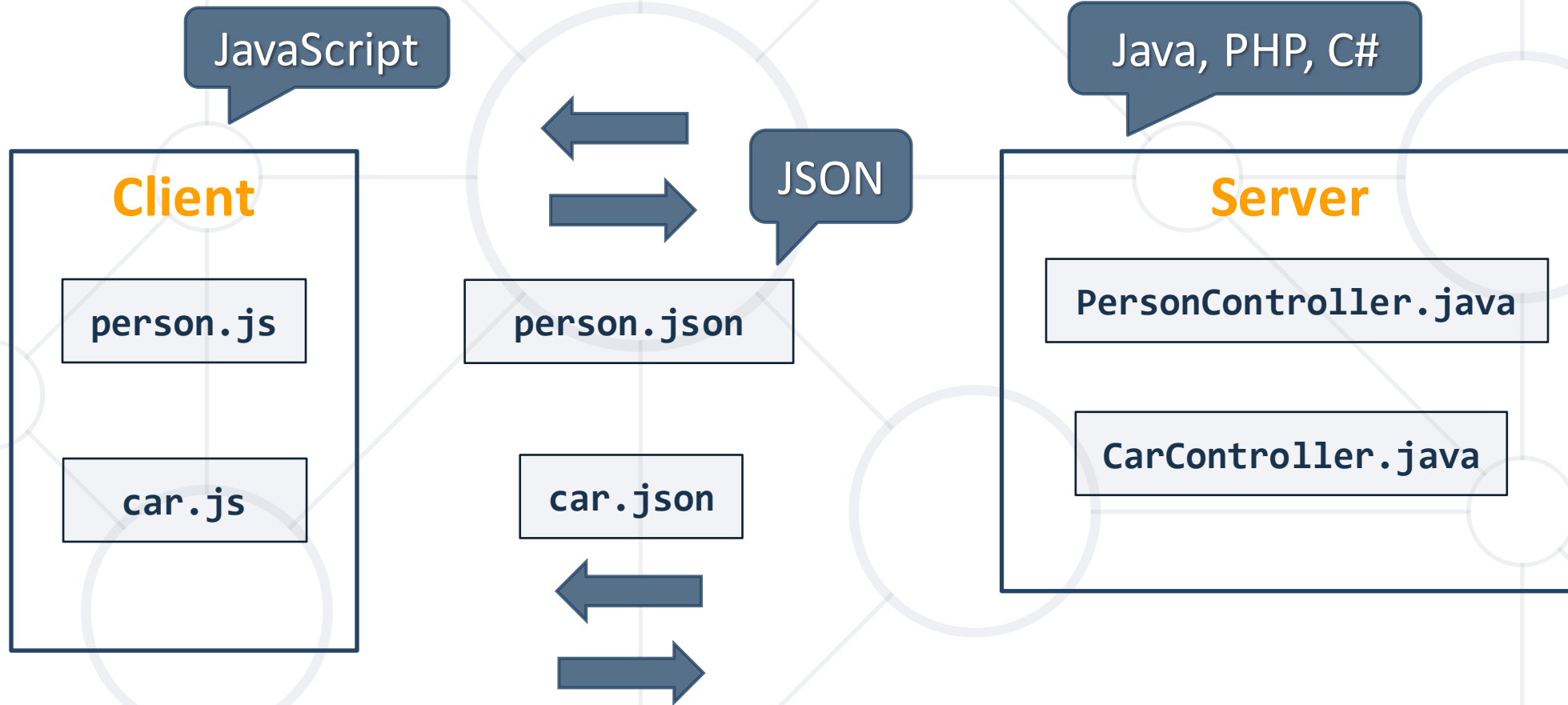
Value

student.json

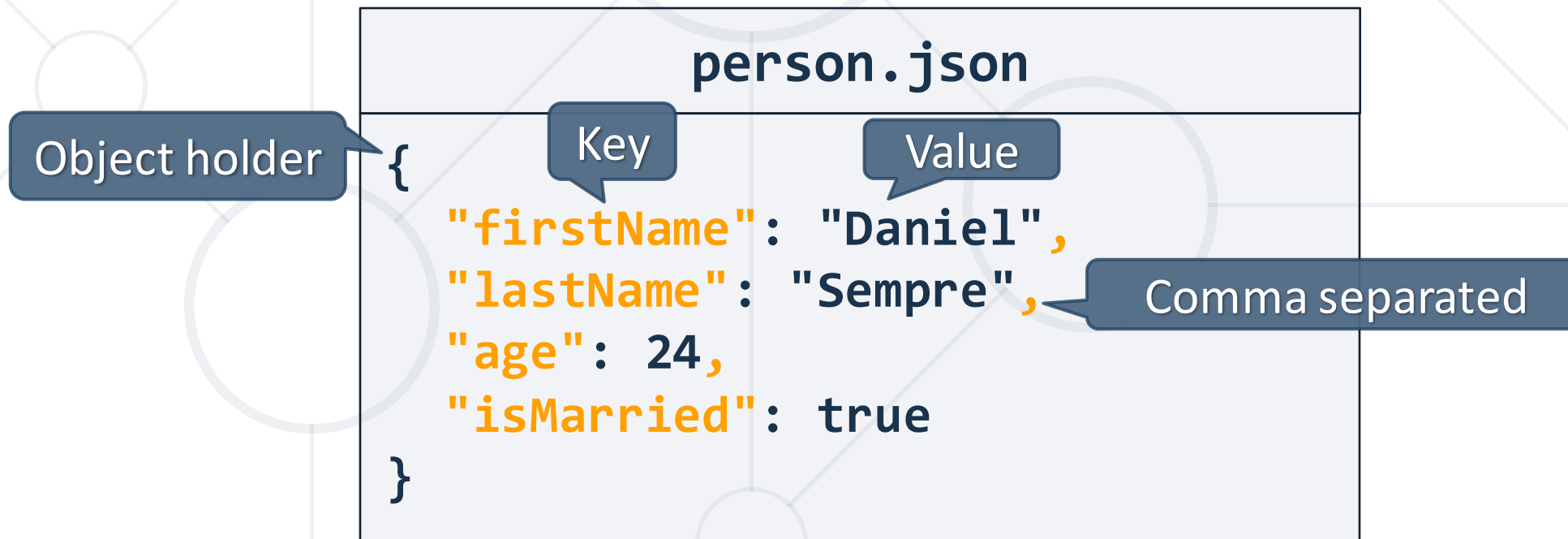
```
{  
  "firstName": "Daniel",  
  "lastName": "Sempere",  
  "age": 24,  
  "courses": [  
    {  
      "name": "Java DB",  
    },  
    {  
      "name": "HTML",  
    },  
  ]  
}
```

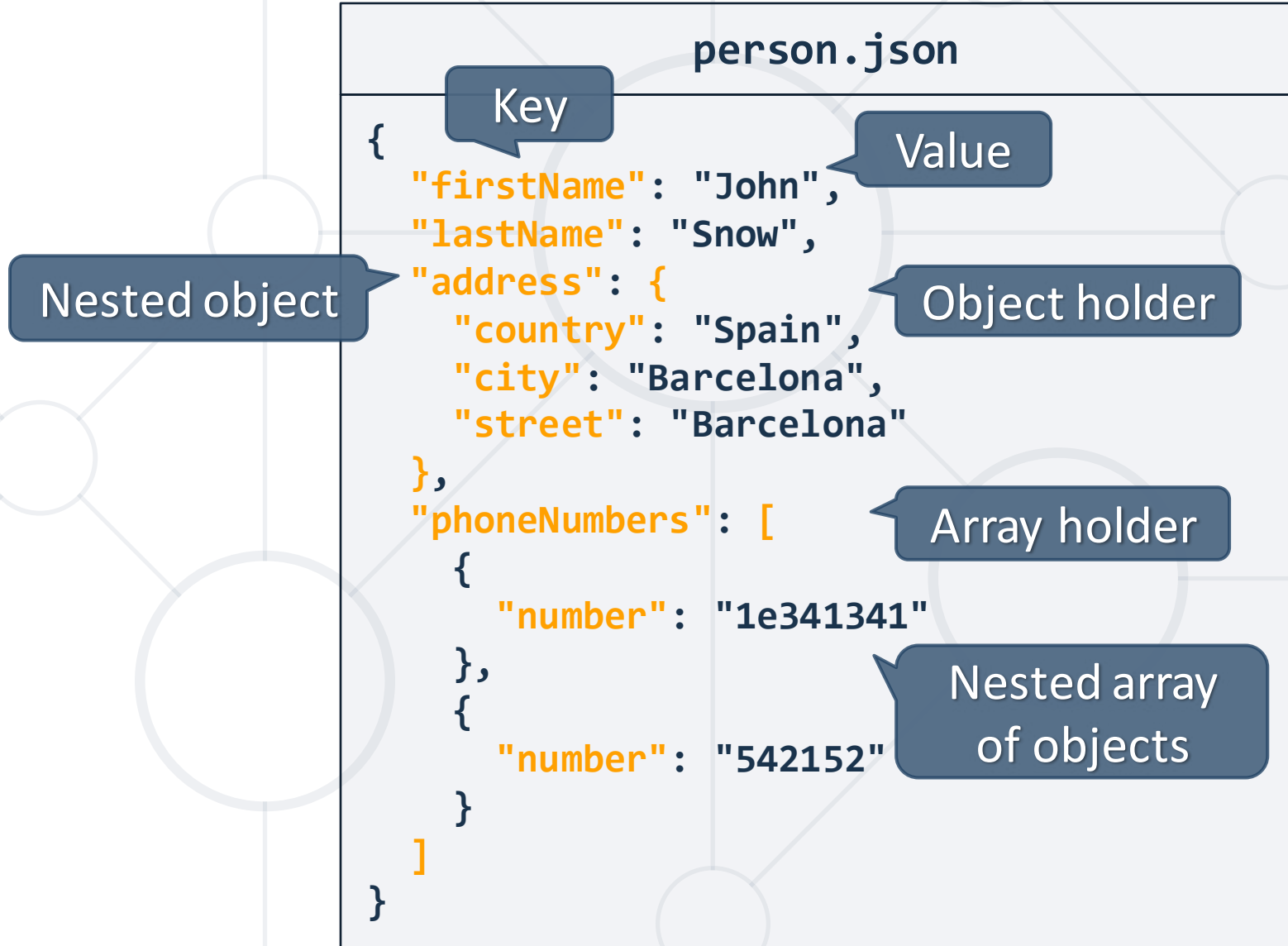
Array type
value

JSON Function



- Data is represented in **name/value** pairs
- Curly braces hold objects
- Square brackets hold **arrays**







GSON

Serialize and De-serialize objects with Java

- Provides easy to use mechanisms to convert **Java to JSON** and vice-versa
 - Originally developed by Google
- Generates compact and readability JSON output

pom.xml

```
<dependency>  
  <groupId>com.google.code.gson</groupId>  
  <artifactId>gson</artifactId>  
</dependency>
```

- Gson objects are responsible for the JSON manipulations
 - GsonBuilder creates an instance of GSON
 - `excludeFieldsWithoutExposeAnnotation()` – excludes fields without **@Expose** annotation
 - `setPrettyPrinting()` – aligns and justifies the created JSON format
 - `create()` – creates an instance of Gson

JsonParser.java

```
Gson gson = new GsonBuilder()  
    .excludeFieldsWithoutExposeAnnotation()  
    .setPrettyPrinting()  
    .create();
```

Export Single Object to JSON

AddressJsonDto.java

```
public class AddressJsonDto implements Serializable {  
  
    @Expose  
    private String country;  
  
    @Expose  
    private String city;  
  
    @Expose  
    private String street;  
  
}
```

The field will be imported/exported

JsonParser.java

```
AddressJsonDto addressJsonDto = new AddressJsonDto();  
    addressJsonDto.setCountry("Bulgaria");  
    addressJsonDto.setCity("Sofia");  
    addressJsonDto.setStreet("Mladost 4");  
String content = this.gson.toJson(addressJsonDto);
```

Creates JSON

Export Single Object to JSON

JsonParser.java

```
AddressJsonDto addressJsonDto = new AddressJsonDto();  
addressJsonDto.setCountry("Bulgaria");  
addressJsonDto.setCity("Sofia");  
addressJsonDto.setStreet("Mladost 4");  
String content = this.gson.toJson(addressJsonDto);
```

address.json

```
{  
  "country": "Bulgaria",  
  "city": "Sofia",  
  "street": "Mladost 4"  
}
```

Export Multiple Object to JSON

JsonParser.java

```
List<AddressJsonDto> addressJsonDtos = new ArrayList<>();  
addressJsonDtos.add(addressJsonDtoBulgaria);  
addressJsonDtos.add(addressJsonDtoSpain);  
String content = this.gson.toJson(addressJsonDtos);
```

addresses.json

```
[  
  {  
    "country": "Bulgaria",  
    "city": "Sofia",  
    "street": "Mladost 4"  
  },  
  {  
    "country": "Spain",  
    "city": "Barcelona",  
    "street": "Las Ramblas"  
  }  
]
```

Import Single Object to JSON

AddressJsonDto.java

```
public class AddressJsonDto implements Serializable {  
  
    @Expose  
    private String country;  
  
    @Expose  
    private String city;  
  
    @Expose  
    private String street;  
  
}
```

The field will be imported/exported

JsonParser.java

```
AddressJsonDto addressJsonDto =  
    this.gson.fromJson(AddressJsonDto.class, "/files/input/json/address.json");
```


Import Single Object to JSON

AddressJsonDto.java

```
public class AddressJsonDto implements  
Serializable {
```

@Expose

```
private String country;
```

@Expose

```
private String city;
```

@Expose

```
private String street;
```

```
}
```

address.json

```
{  
  "country": "Bulgaria",  
  "city": "Sofia",  
  "street": "Mladost 4"  
}
```

Import Multiple Object to JSON

JsonParser.java

```
AddressJsonDto[] addressJsonDtos =  
    this.gson.fromJson(AddressJsonDto[].class, "/files/input/json/addresses.json");
```

Object Array

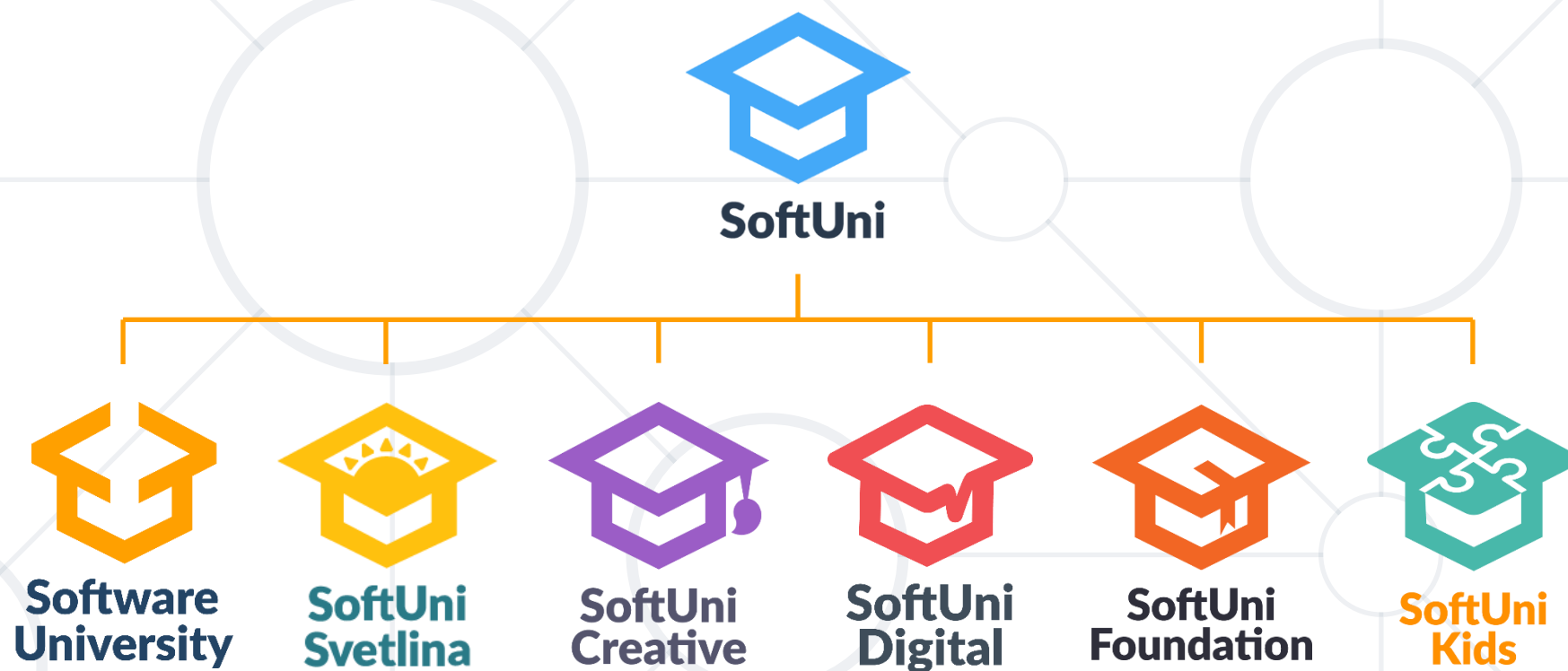
addresses.json

```
[  
  {  
    "country": "Bulgaria",  
    "city": "Sofia",  
    "street": "Mladost 4"  
  },  
  {  
    "country": "Spain",  
    "city": "Barcelona",  
    "street": "Las Ramblas"  
  }  
]
```

- JSON is a very easy to use and understand format
- GSON is a java library to operate with JSON files
 - Easy import and export



Questions?



SoftUni Diamond Partners



XSsoftware



SBTech
we know sports



telenor



SoftwareGroup
doing it right

NETPEAK



SmartIT



Postbank

Решения за твоето утре

**SUPER
HOSTING
.BG**

INDEAVR

Serving the high achievers



INFRAGISTICS®



STEMO®
Computer Systems & Software



SoftUni Organizational Partners



OneBit
SOFTWARE



WORLD
OF
MYTHS

- Software University – High-Quality Education, Profession and Job for Software Developers
 - softuni.bg, softuni.org
- Software University Foundation
 - softuni.foundation
- Software University @ Facebook
 - facebook.com/SoftwareUniversity
- Software University Forums
 - forum.softuni.bg



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://softuni.org>
- © Software University – <https://softuni.bg>

