selsoft
build better, deliver faster

Developing RESTful Web Services with Java

4

Chapter 4: JSON Processing

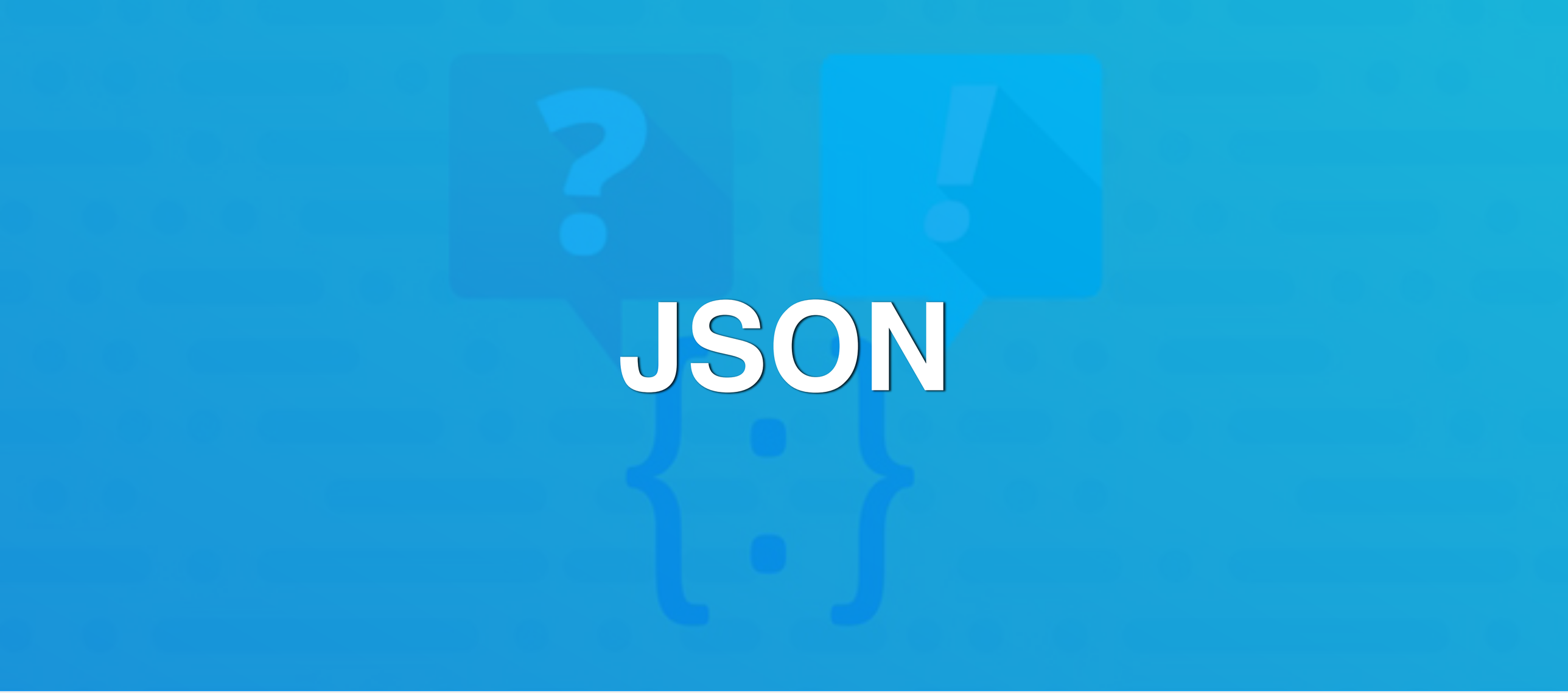
Eğitmen:

Akın Kaldıroğlu

Çevik Yazılım Geliştirme ve Java Uzmanı

Topics

- **JSON**
- **JSON Support in Java EE**
- **JSON Processing**
  - Object Model
  - Streaming Model
- **JSON Binding**



selsoft

build better, deliver faster

JSON - I

- **JSON** stands for **JavaScript Object Notation**.
- It is a lightweight data-interchange format (https://www.json.org/json-en.html).
- It is language-independent although it’s ben concieved in JavaScript (JS) world.
- JSON is schema-less.

JSON - II

- It is simpler and lighter than XML, which is the first, common, language-independent data-interchange format.
- So it can be thought of a replacement of XML.
- Many databases relational (PostgreSQL and MySQL) or NoSQL (MongoDB and Neo4j) have support to store and query JSON.

JSON Structures

- JSON has only two data structures: **object** and **array**
  - Object is a unordered collection of name/value pairs within a pair of braces such as `{"name":"Zeynep"}`.
    - It is represented as classes in OOP languages.
  - Array is an ordered list of values within a pair of brackets such as `[{"name" : "Zeynep"}, {"name" : "Kerim"}]`.
    - It is represented as arrays or lists in languages.

persons.json

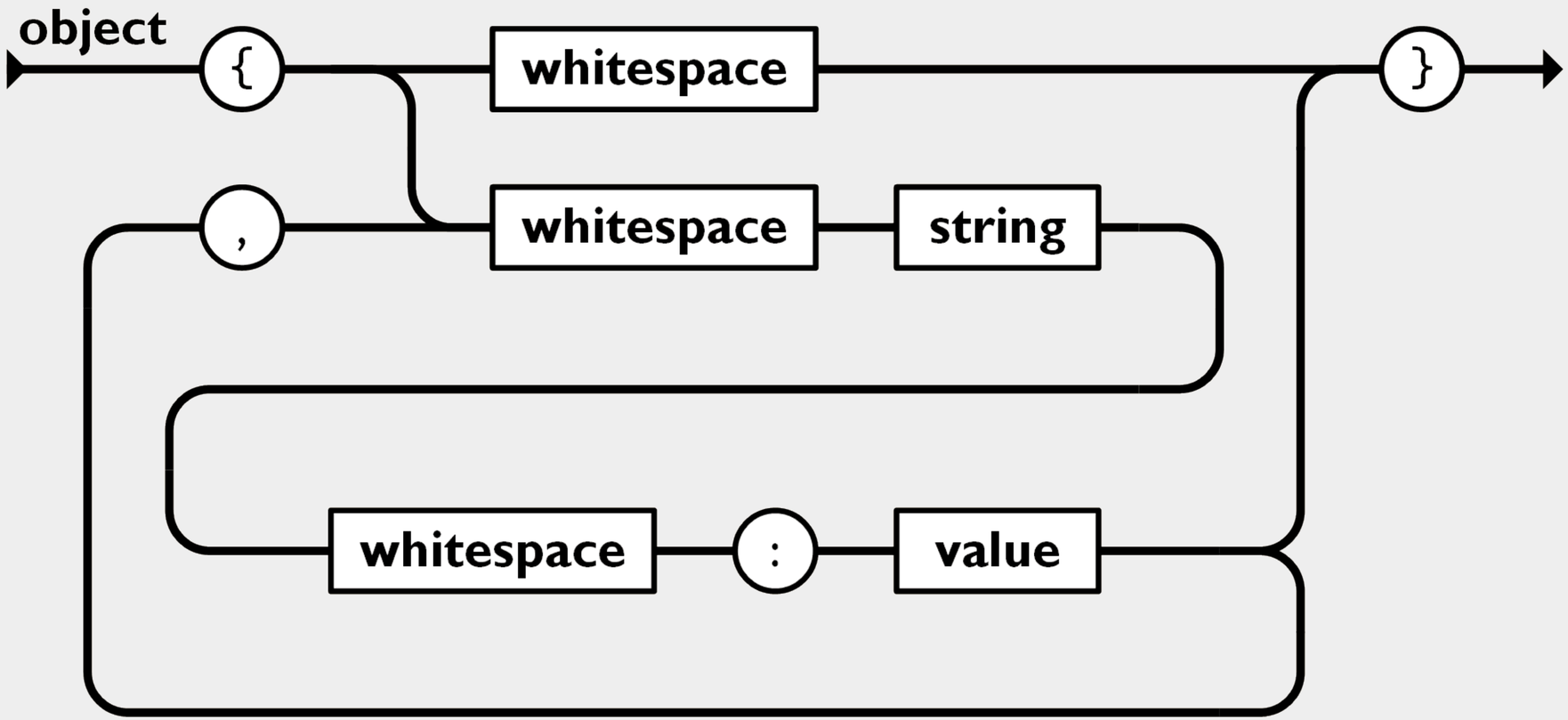
- **JSON Processing** project.
- `person.json`.

Object - I

- A JSON object begins with `{` and ends with `}`.
- The name and value in a pair are separated by a colon `:`.
- If there are more than one name-value pairs in an object they are separated by a comma `,`.
- Names are strings and must be in a double quote such as **`“name”`**.
- But values may be of any of the seven value types (string, number, object, array, true, false, and null), including another object or an array.

Object - II

- An object is an unordered set of name/value pairs.



persons.json

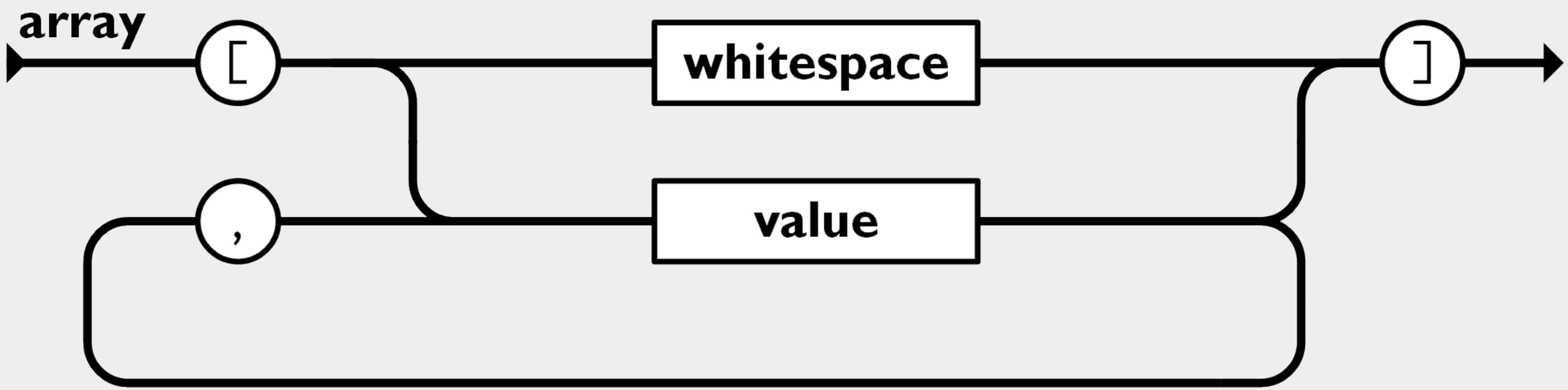
- **JSON Processing** project.
- `person.json`.

Array - I

- Arrays are enclosed in brackets `[]`.
- Array is an ordered list of values and their values are separated by a comma `,`.
- Each value in an array may be of a different type, including another array or an object.

Array - II

- An array is a list of ordered values.



persons.json

- **JSON Processing** project.
- `persons.json`.

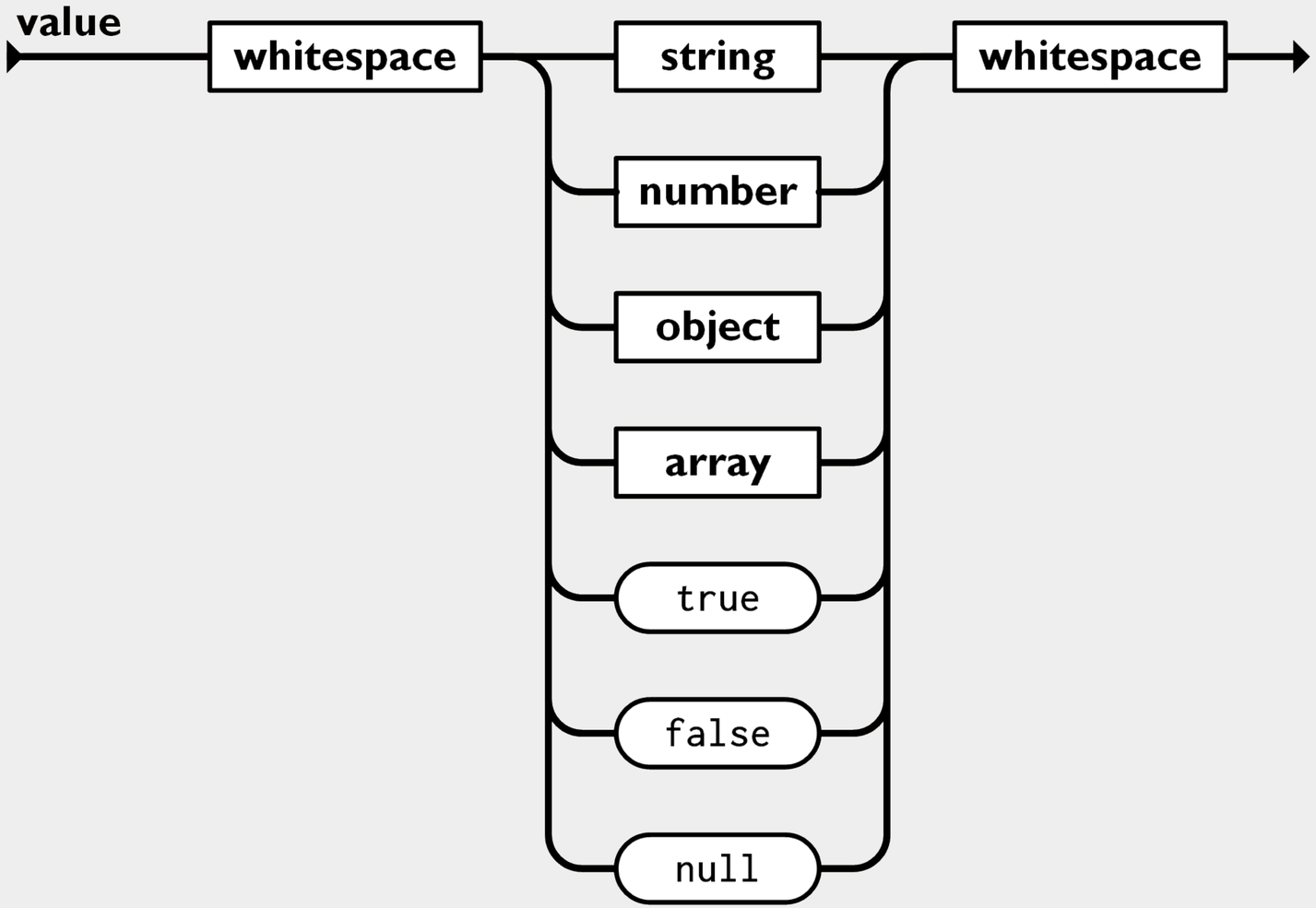
JSON Types

- JSON has types other than object and array:
  - `string`
  - `number`
  - `true`
  - `false`
  - `null`

Value - I

- A value can be one of those seven types.
- These structures can be nested.

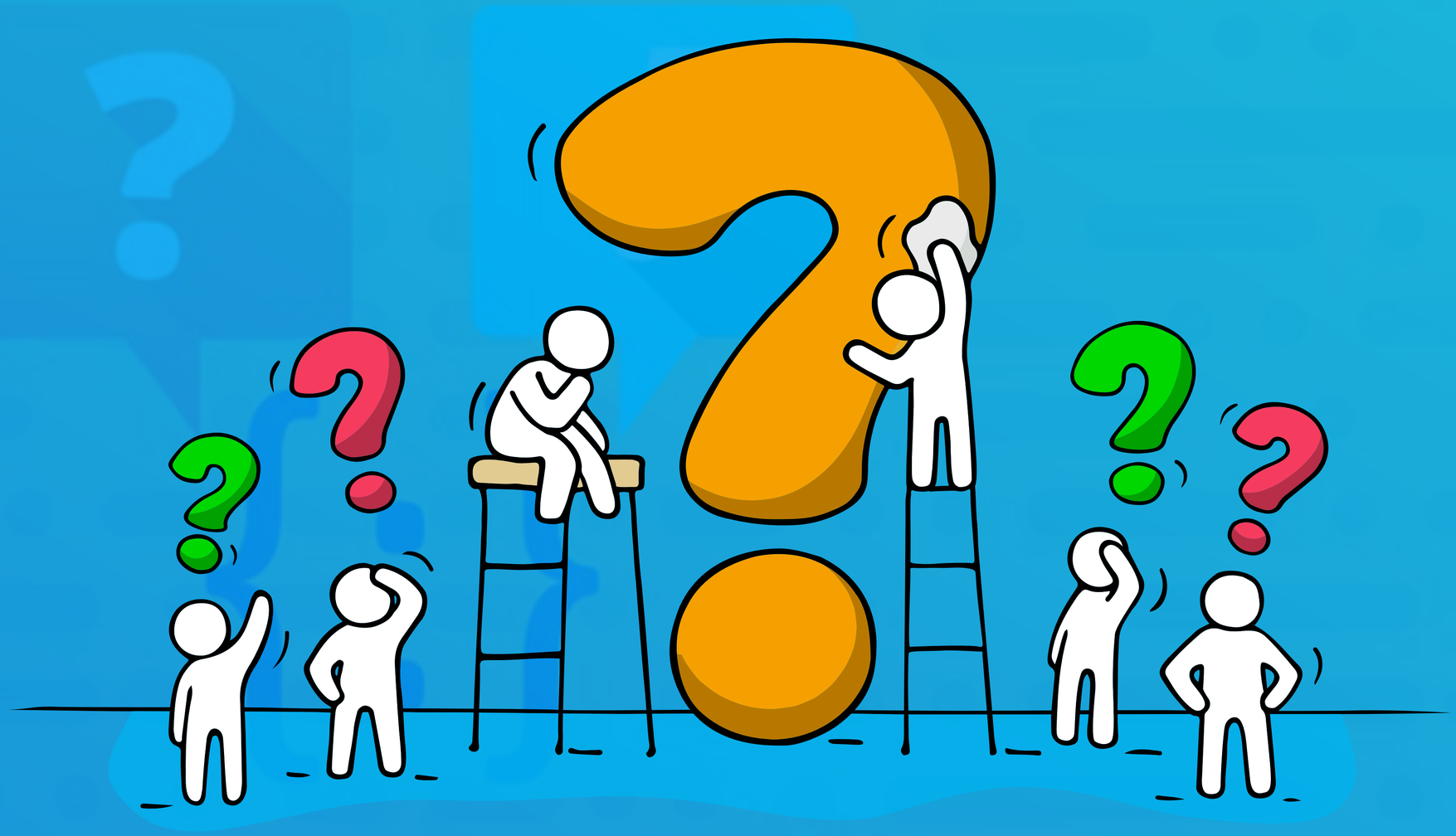
Value - II


value
whitespace
string
number
object
array
true
false
null
whitespace

Limitations of JSON

- JSON has limitations too:
  - No schema.
  - It has no date type.
  - It has only one number type which is double-precison.
  - No comments.
  - And some think that JSON is still verbose and may prefer more concise formats than JSON.

Time for questions!



selsoft
build better, deliver faster

info@selsoft.com.tr
selsoft.com.tr

JSON Support in Java EE

selsoft

build better, deliver faster

JSON Support in Java EE 8 - I

- Java EE 8 has support for both JSON processing and binding.
  - JSON processing (JSONP) is about parsing existing JSON documents and creating new ones.
  - JSON binding is about relationship between JSON documents and Java classes.
- Java EE 7 has support only for JSON processing through JSR-353.

JSON Support in Java EE 8 - II

- Java EE 8 has a support for JSON processing in **JSR 374**, successor of JSR-353
  - JSR 374 is **Java API for JSON Processing 1.1 (JSON-P)** (https://www.jcp.org/en/jsr/detail?id=374) which provides an API to parse, transform, and query JSON data using the object model or the streaming model.
- Java EE 8 has a support for JSON binding in **JSR 367**.
  - JSR 367 is **JAVA API for JSON Binding (JSON-B)** (https://jcp.org/en/jsr/detail?id=367) which provides a standard binding layer (metadata and runtime) between Java classes and JSON documents.

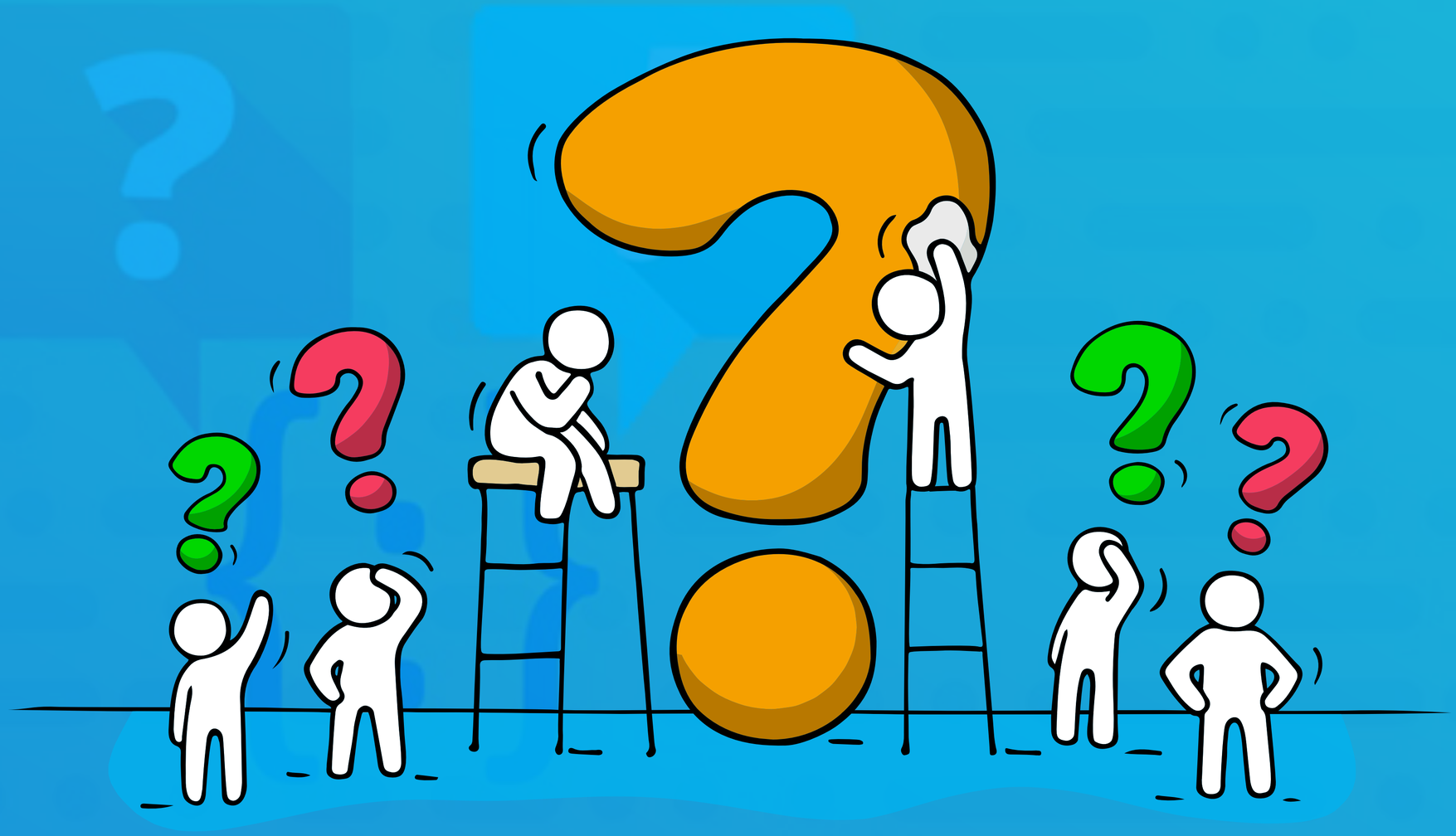
JSON Support in Java EE 8 - III

- GlassFish has the reference implementation for JSON-P available at https://javaee.github.io/jsonp/.
- The JSON-B reference implementation is **Yasson**, which is developed through Eclipse.org and is included as part of **GlassFish**.
  - More about Yasson at https://projects.eclipse.org/projects/rt.yasson.
  - It is available at https://javaee.github.io/jsonb-spec/.

JSON Support in Java EE 8 - VI

- Java EE 8 Tutorial for JSON: https://javaee.github.io/tutorial/jsonp002.html.
- Java EE 8 Tutorial for JSON-B: https://javaee.github.io/tutorial/jsonb.html
- Short Java EE 8 Tutorial for JSON: https://javaee.github.io/tutorial/jsonp001.html

Time for questions!



selsoft
build better, deliver faster

info@selsoft.com.tr
selsoft.com.tr

JSON Processing

selsoft

build better, deliver faster

JSON Processing - I

- The Java API for JSON Processing has the following packages:
  - **`javax.json`**: Provides an object model API to process JSON.
  - **`javax.json.stream`**. Provides a streaming API to parse and generate JSON.
  - **`javax.json.spi`**. Service Provider Interface (SPI) to plug in implementations for JSON processing objects.

JSON Processing - II

- So JSON processing there are mainly two models:
  - **Object model** where whole JSON document is read into memory as a tree.
  - **Streaming model** where whole JSON document isn't read into memory but in blocks.
    - It is event based and the applications listens to the events fired by the parser.

selsoft

build better, deliver faster

Object Model

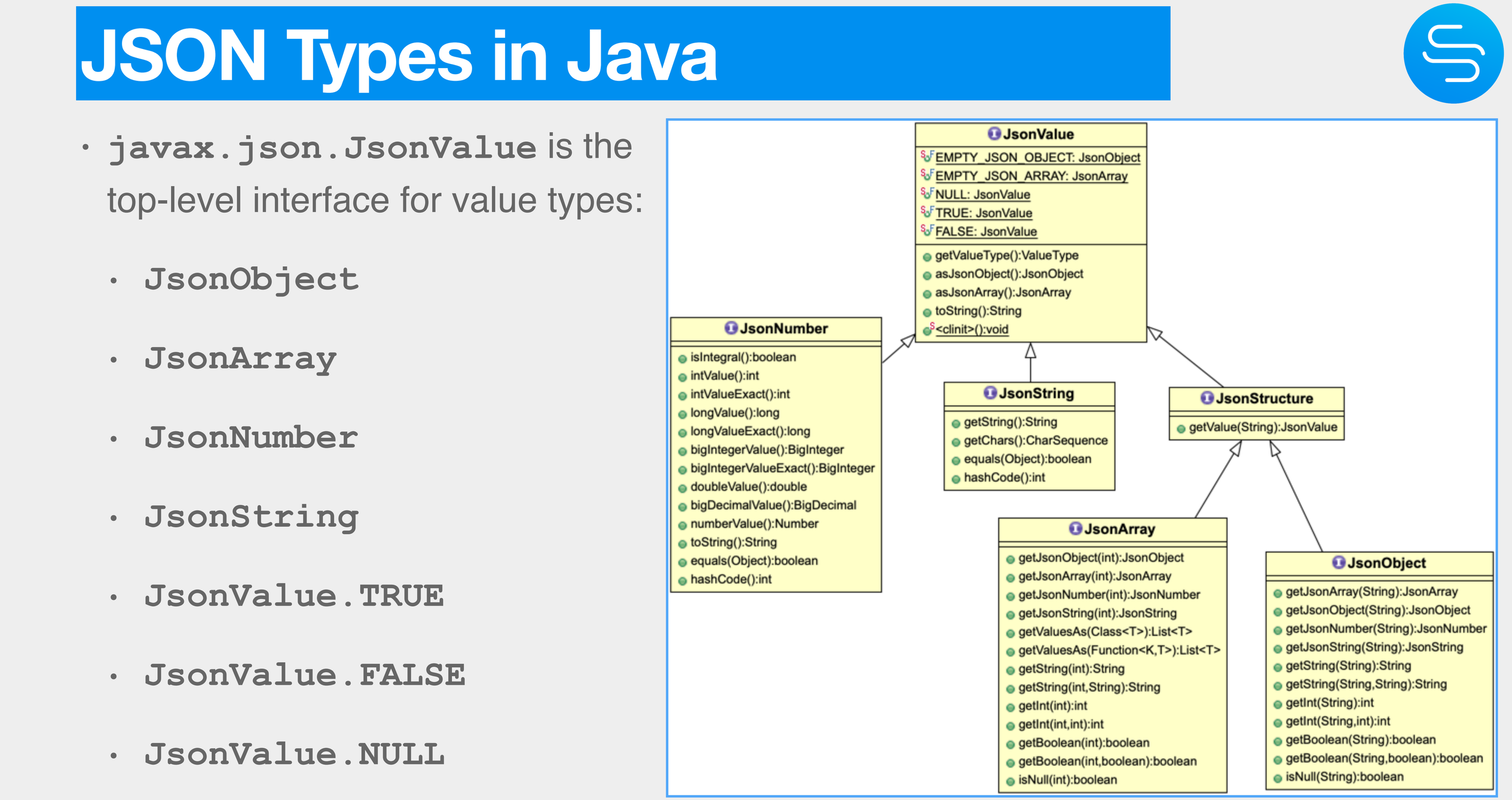
Object Model for JSON

- In object model whole JSON document is read into memory as a tree.
- This is similar to Document Object Model (DOM) of XML.
- It provides a simple and intuitive way to read and process documents.
- But it requires more memory which can be a problem with large documents because it reads whole document into the memory.
- Object model allows both reading and writing JSON documents.

Main Objects of Object Model

- Main object in `javax.json` package mainly for processing a JSON document as an object model.
  - `Json`
  - `JsonReader`
  - `JsonWriter`
  - `JsonObjectBuilder`
  - `JsonArrayBuilder`
  - `JsonValue`

JSON Types in Java



- `javax.json.JsonValue` is the top-level interface for value types:
  - `JsonObject`
  - `JsonArray`
  - `JsonNumber`
  - `JsonString`
  - `JsonValue.TRUE`
  - `JsonValue.FALSE`
  - `JsonValue.NULL`

Json Class

- **`javax.json.Json`** is the unique class in the package.
- It is a factory class for creating JSON processing objects through static methods.
- This class provides the most commonly used methods for creating objects such as **`JsonReader`**, **`JsonWriter`**, **`JsonObjectBuilder`**, etc.
- It also creates factories for those obejcts.
  - The factory classes provide all the various ways to create these objects.

JsonReader

- To read a JSON document in object model from a source
  - First a **`JsonReader`** object is created using **`Json`** and
  - Then suitable **`JsonStructure`** object i.e. either **`JsonObject`** or **`JsonArray`** is read and
  - Then processing goes on by reading following structures.

PersonReader

- **JSON Processing** project.
- `org.javaturk.json.objectModel.PersonReader`

JsonBuilder

- To build a JSON document in object model
  - First a **`JsonObjectBuilder`** or **`JsonArrayBuilder`** object is created and,
  - Then using **`add()`** methods in the fashion of builder pattern, values are added,
  - And at the end the object is build by calling **`build()`** method.

PersonBuilder

- **JSON Processing** project.
- `org.javaturk.json.objectModel.parsing.PersonBuilder`

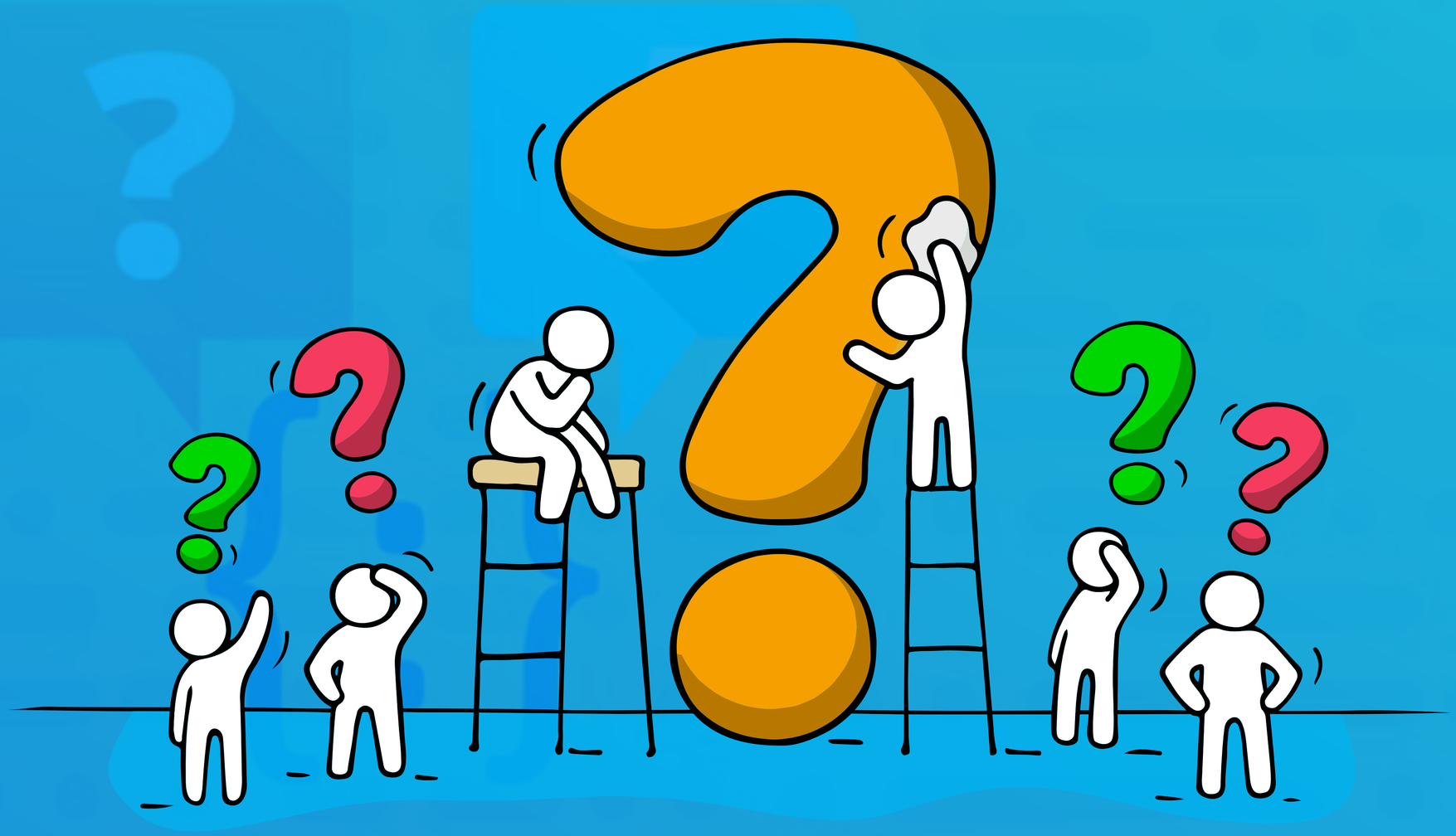
JsonWriter

- To write a JSON document in object model:
  - First a **`JsonWriter`** object is created and
  - Then one of following **`write`** methods is called
    - **`write(JsonStructure)`**, **`write(JsonValue)`**
    - **`writeObject(JsonObject)`**, **`writeArray(JsonArray)`**

PersonWriter

- **JSON Processing** project.
- `org.javaturk.json.objectModel.parsing.PersonWriter`

Time for questions!



selsoft
build better, deliver faster

info@selsoft.com.tr
selsoft.com.tr

selsoft

build better, deliver faster

Streaming Model

Streaming Model for JSON - I

- Streaming model is similar to SAX (Simple API for XML) for XML.
- It is event-based parser and the applications listens to the events fired by the parser.
- In this model where whole JSON document isn’t read into memory.
- The parser reads the document in blocks and fires events when it encounter different notes of the document.
- So it doesn’t require more memory which makes a perfect fit for larger documents.

Streaming Model for JSON - II

- It allows both reading and writing JSON documents.
- Generally it is more efficient than object model to process JSON documents.

Main Objects of Streaming Model

- Main objects of streaming model are in **`javax.json.stream`** package:
  - **`JsonParser`**: It provides a read-only access to JSON documents.
  - **`JsonParser.Event`**: It is an enum and defines event types for JSON documents.
  - **`JsonGenerator`**: It writes JSON data to an output source in a streaming way.
- Both of them are created by **`Json`**.

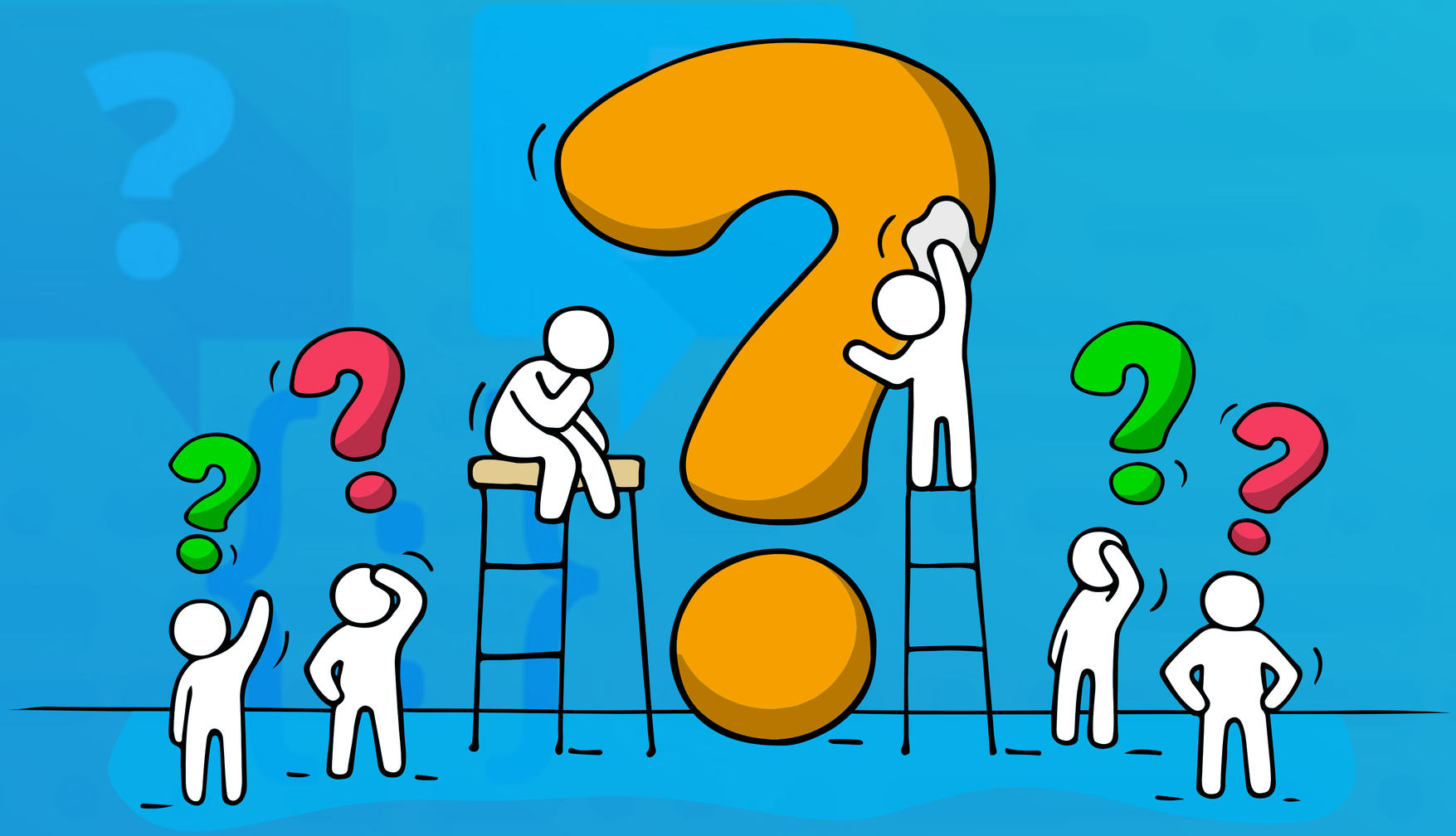
GenericJSONParser

- **JSON Processing** project.
- `org.javaturk.json.streamModel.GenericJSONParser`

PersonGenerator

- **JSON Processing** project.
- `org.javaturk.json.streamModel.PersonGenerator`

Time for questions!



selsoft
build better, deliver faster

info@selsoft.com.tr
selsoft.com.tr

JSON Binding

selsoft

build better, deliver faster

JSON Binding

- JSON-B is about serializing Java objects to JSON documents and deserializing JSON documents into Java objects.
- Java EE 8 has a support for JSON binding in **JSR 367**.
  - JSR 367 is **JAVA API for JSON Binding (JSON-B)** (https://jcp.org/en/jsr/detail?id=367) which provides a standard binding layer (metadata and runtime) between Java classes and JSON documents.

Packages

- Packages for JSON-B in Java EE are:
  - `javax.json.bind package`
  - `javax.json.bind.adapter`
  - `javax.json.bind.annotation`
  - `javax.json.bind.config`
  - `javax.json.bind.serializer`
  - `javax.json.bind.spi`

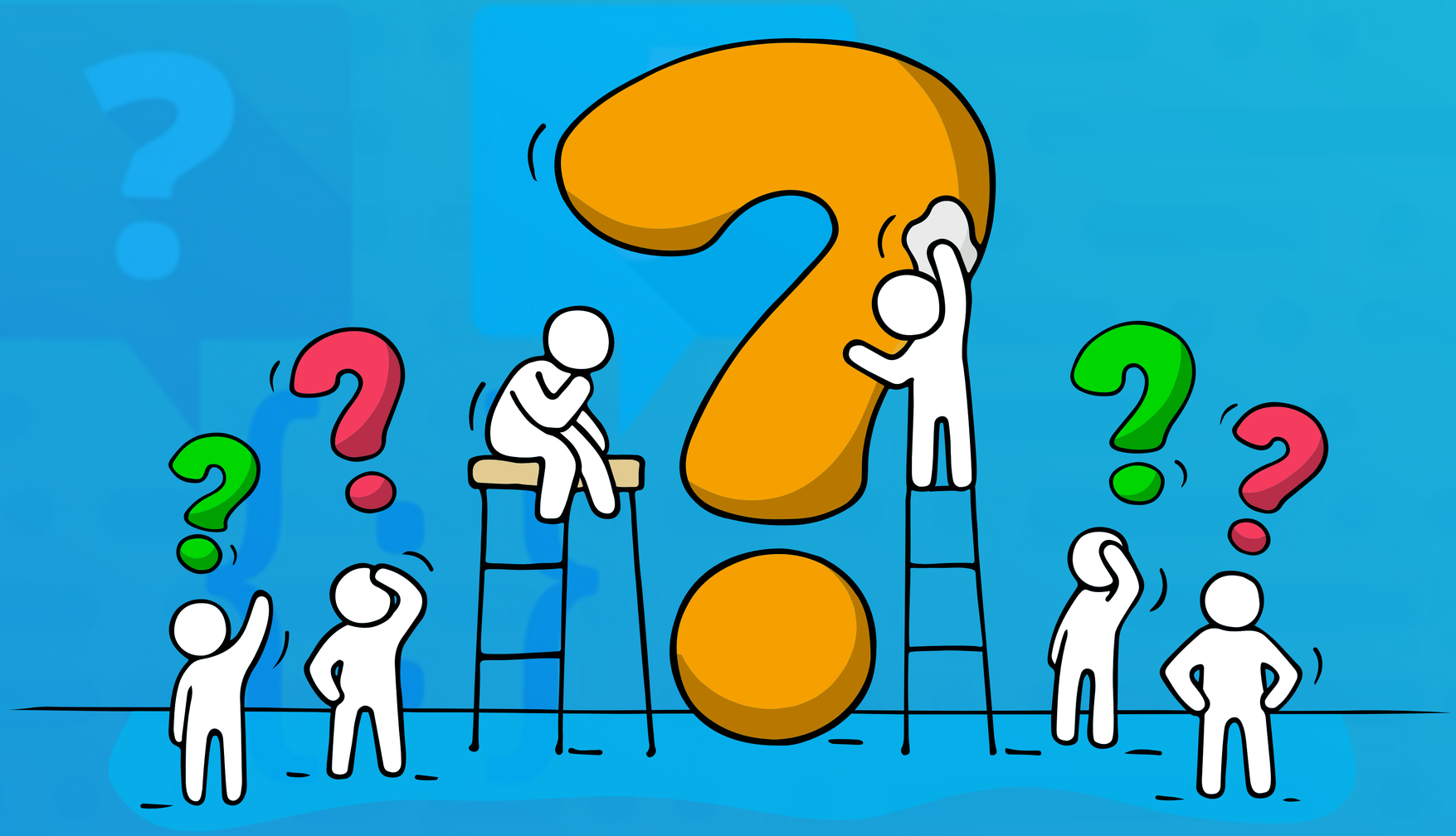
Main Objects

- Main objects for JSON-B are:
  - **`Jsonb`**: It has methods for serializing Java objects to JSON and deserailizing JSON to Java objects.
  - **`JsonbBuilder`**: It is used to create **`Jsonb`** instances.
  - **`JsonbConfig`**: It is used to configure **`Jsonb`** instances.
  - **`JsonBException`**: It is thrown when a problem during binding happens.

PersonBinding

- **JSON Processing** project.
- `org.javaturk.json.binding.PersonBinding`

Time for questions!



selsoft
build better, deliver faster

info@selsoft.com.tr
selsoft.com.tr

Other Libraries

selsoft

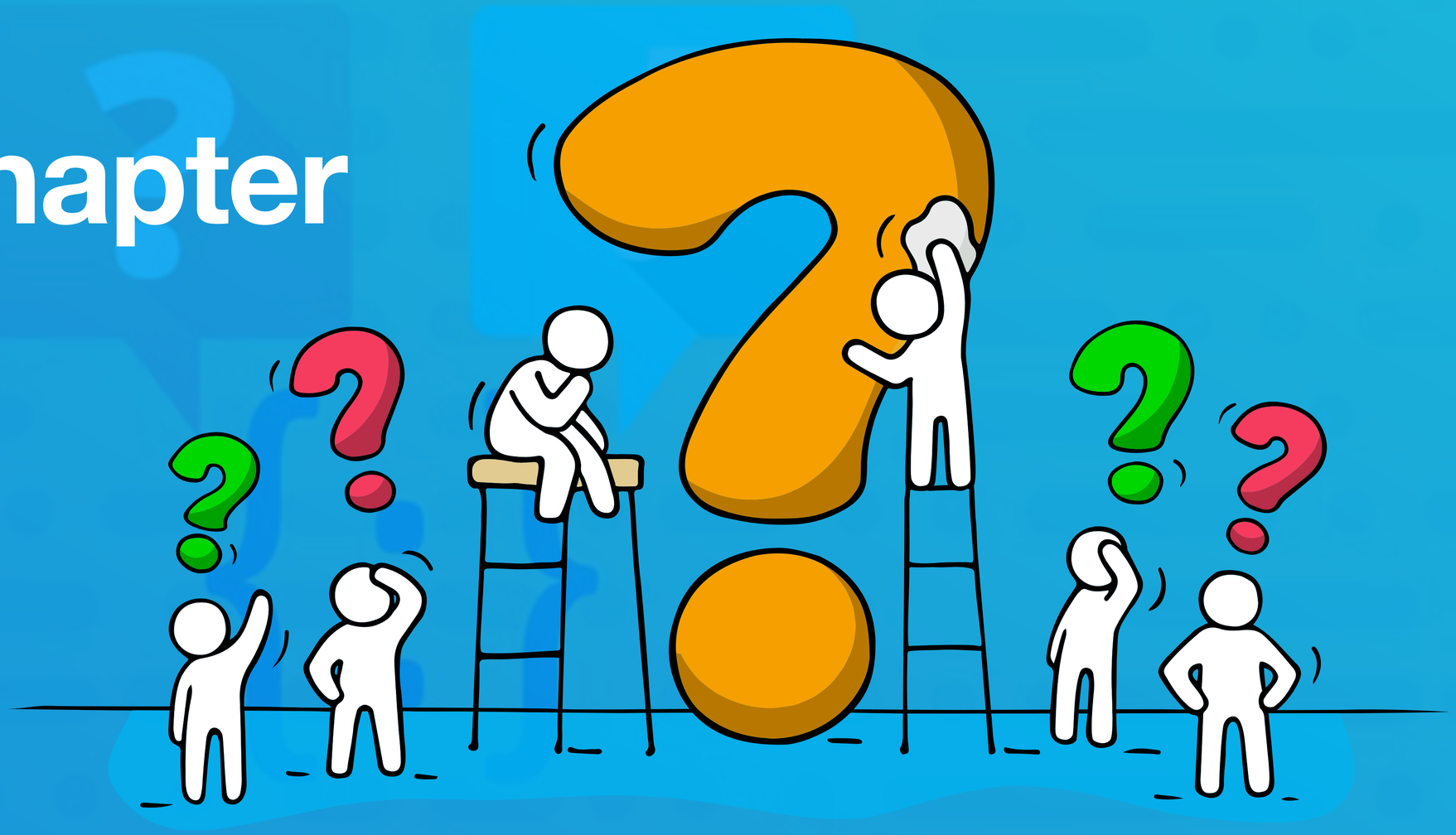
build better, deliver faster

Third-Party Libraries

- There are other third-party libraries to process JSON in Java world:
  - **Jackson**: It provides both object, streaming model and data binding for JSON.
    - https://github.com/FasterXML/jackson
  - **GSON**: It is a library to serialize/deserializa between JSON and Java object.
    - It is originally developed by Google and later open-sourced.
    - https://github.com/google/gson

End of Chapter

Time for Questions!



selsoft
build better, deliver faster

info@selsoft.com.tr
selsoft.com.tr