JSON Manipulation Library

Advanced Programming - Project 2024/2025

The aim is to create a Kotlin library for generating and manipulating JSON using in-memory models, involving all its values types:

- Objects
- Arrays
- Strings
- Numbers
- Booleans
- Null

> **Note:** The library will not handle *parsing* of textual JSON (although this would make sense as a library component). The focus is only on in-memory manipulation and further serialization to strings that adhere to the standard format.

> **Attention:** No external libraries/frameworks are allowed in the development, except for JUnit.

JSON Model (Phase 1)

Goal: Develop classes to represent JSON values (model), allowing manipulation operations and serialization to strings (standard format).

The API of the library classes must allow programmers to:

- Instantiate JSON values programmatically, creating and composing objects from the library classes;
- Perform filtering that produces new JSON Objects or Arrays without changing existing ones:
  - JSON Object → JSON Object
  - JSON Array → JSON Array
- Perform mapping operations on Arrays (*map*):
  - JSON Array → JSON Array
- Use *visitors*, to facilitate the developed of new features that involve traversing the structure recursively. This characteristic should be illustrated with functionalities to:
  - Validate whether all JSON objects are valid (keys with valid content and unique);
  - Check if all JSON Arrays contain values of the same type (not Null)
- Serialize a model to a string, ensuring compatibility with the standard. (Formatting issues are not very relevant, as long as the output strings are valid JSON).

Inference (Phase 2)

Goal: Develop a function that allows programmers to instantiate the JSON Model (Phase 1) from Kotlin objects.

This feature should be based on reflection, and it is only required to support the following Kotlin types:

- Int
- Double
- Boolean
- String
- List< *supported type* >
- Enums
- null
- data classes with properties whose type is supported
- maps (Map) that associate Strings (keys) to any of the above Kotlin elements

The following presents an example scenario, where the object referenced by *course* would lead to the JSON structure on the right, presented in serialized format for convenience.

```kotlin
data class Course(
    val name: String,
    val credits: Int,
    val evaluation: List<EvalItem>
)

data class EvalItem(
    val name: String,
    val percentage: Double,
    val mandatory: Boolean,
    val type: EvalType?
)

enum class EvalType {
    TEST, PROJECT, EXAM
}
```

```kotlin
val course = Course(
    "PA", 6, listOf(
        EvalItem("quizzes", .2, false, null),
        EvalItem("project", .8, true,
EvalType.PROJECT)
    )
)
```

```json
{
 "name": "PA",
 "credits": 6,
 "evaluation": [
   {
     "name": "quizzes",
     "percentage": 0.2,
     "mandatory": false,
     "type": null
   },
   {
     "name": "project",
     "percentage": 0.8,
     "mandatory": true,
     "type": "PROJECT"
   }
 ]
}
```

(example of serialized model inferred from course)

Attention: This feature should instantiate the model (Phase 1), instead of concatenating JSON strings directly. The advantage of this is to be able to post process the JSON in memory, e.g., filtering elements, etc.

General Requirements

- The library API must be documented using KDoc;
- All features must have corresponding JUnit tests;
- The library source should be published in a GitHub repository, including:
    - A tutorial on how to use the library
    - A UML class diagram that represents the classes that model JSON
    - A release (JAR)

(Phase 3 - To Be Defined)