7

GSON —> JsonReader

1. Clase `JsonReader` de GSON

- **Descripción**: `JsonReader` es un analizador JSON en streaming de GSON que permite leer JSON como una secuencia de tokens (`JsonToken`), recorriendo los elementos JSON en profundidad y orden secuencial en el mismo orden en que aparecen en el elemento JSON.
- **Tipos de Elementos**:
    - Literales: cadenas, números, booleanos, nulos.
    - Delimitadores de objetos y arrays: `{`, `}`, `[`, `]`.
- **Tipos de Analizadores**:

- **Pull Parser**: analizador en el que el código que lo utiliza **extrae los tokens del analizador cuando está listo para gestionar el siguiente token.** `JsonReader` es de este tipo.
- **Push Parser**: analiza los tokens JSON y los envía a un gestor de eventos.

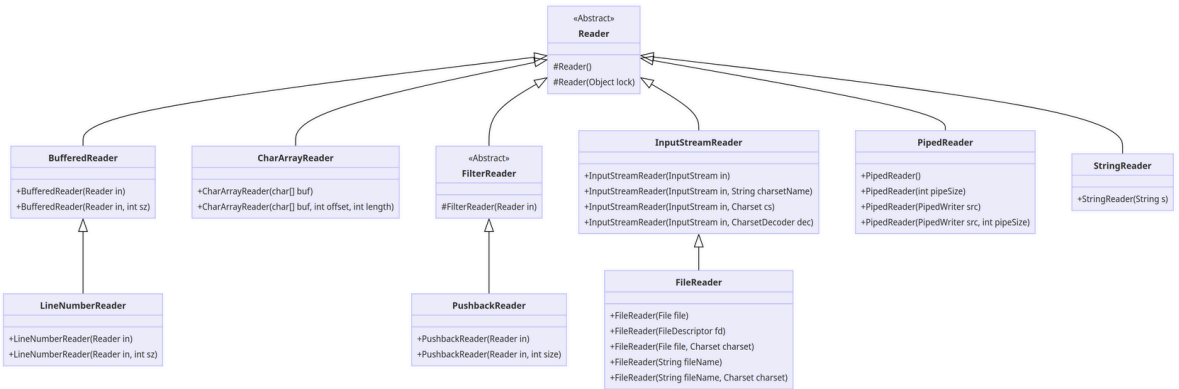
1.1 Creación de un `JsonReader`

- Se crea con un `Reader` de Java:

```
JsonReader jsonReader = new JsonReader(new StringReader(jsonString));
```

- El `StringReader` convierte una cadena Java en una secuencia de caracteres.

⚠️ `Readers` que existen en Java. Entre otros: `BufferedReader` **(y subclase** `LineNumberReader` **),** `CharArrayReader` **,** `FilterReader` **(subclase** `PushbackReader` **),** `InputStreamReader` **(y subclase** `FileReader` **),** `PipedReader` **o** `StringReader`



---

2. Iteración de Tokens `JsonToken`

- `JsonReader` recorre tokens con un bucle que verifica `hasNext()` para seguir avanzando.
- **Tipos de Tokens (`JsonToken`):**

Constante enumeración	Descripción
BEGIN_ARRAY	Apertura de un array JSON.
BEGIN_OBJECT	Apertura de un objeto JSON.
BOOLEAN	Valor JSON `true` o `false`.
END_ARRAY	Cierre de un array JSON.
END_DOCUMENT	Final del flujo JSON.
END_OBJECT	Cierre de un objeto JSON.
NAME	Nombre de una propiedad JSON.
NULL	Valor JSON nulo.
NUMBER	Número JSON representado por un *double*, *long* o *int* en Java.
STRING	String JSON.

- Ejemplo de iteración con `if` y `switch` para leer diferentes tokens:

```
String json = "{\"nome\" : \"Alejandra Pizarnik\", \"idade\" : 36}";

JsonReader jsonReader = new JsonReader(new StringReader(json));

try {
    while (jsonReader.hasNext()) {
```

```
        JsonToken siguienteToken = jsonReader.peek(); // de
vuelve el siguiente, sin consumirlo.
        System.out.println(siguienteToken);

        if (JsonToken.BEGIN_OBJECT == siguienteToken) {
            // Si es un objeto, consumimos las llaves {
            jsonReader.beginObject();

        } else if (JsonToken.NAME == siguienteToken) {
            // Si es un nombre de atributo, lo imprimimos.
            String nomeAtributo = jsonReader.nextName();
            System.out.println(nomeAtributo);

        } else if (JsonToken.STRING == siguienteToken) {
            // si es una cadena, recuperamos String y la impri
mimos
            String valorString = jsonReader.nextString();
            System.out.println(valorString);

        } else if (JsonToken.NUMBER == siguienteToken) {
            // Si es un número, OJO con los tipos...
            long valorNumero = jsonReader.nextLong();
            System.out.println(valorNumero);

        }
    }
} catch (IOException e) {
    System.err.println(e.getMessage());
}
```

```
String json = "{\"nome\" : \"Alejandra Pizarnik\", \"idade\" : 36

JsonReader jsonReader = new JsonReader(new StringReade

while (jsonReader.hasNext()) {
    JsonToken siguienteToken = jsonReader.peek(); // devuelv
    System.out.println(siguienteToken);

    if (null != siguienteToken) {
        switch (siguienteToken) {
            case BEGIN_OBJECT -> // Si es un objeto, consumimo
                jsonReader.beginObject();
            case NAME -> {
                // Si es un nombre de atributo, lo imprimimos.
                String nomeAtributo = jsonReader.nextName();
                System.out.println(nomeAtributo);
            }
            case STRING -> {
                // si es una cadena, recuperamos String y la imprim
                String valorString = jsonReader.nextString();
                System.out.println(valorString);
            }
            case NUMBER -> {
                // Si es un número, OJO con los tipos...
                long valorNumero = jsonReader.nextLong();
                System.out.println(valorNumero);
            }
            default -> {
            }
        }
```

```
    }
}
```

⚠️ El **método** `peek()` **del** `JsonReader` **devuelve el siguiente token JSON, pero sin moverse sobre él** (sin devolver el siguiente). Sucesivas llamadas a `peek()` devolverán el mismo token JSON.

---

3. Parser Personalizado de JSON con `JsonReader`

- **Objetivo**: Crear un parser personalizado que maneje JSON complejo, como arrays de objetos JSON.
- **Método Principal**:
  - Crea el `JsonReader` a partir de un `InputStream` y llama a un método que recorre los tokens del JSON.

```
public class PoemaJsonReader {

    // Método principal de entrada
    public List<Poema> readJsonStream(InputStream in) throws I
        // JsonReader necesita un Reader, por lo que convertimos e
        // Además, implanta la interfaz Closeable, por lo que podem
        try(JsonReader reader = new JsonReader(new InputStream
            return readArrayPoemas(reader);
        }
```

```
    }
}
```

- **Métodos de Control**:
  - Arrays: Consumen `beginArray()`, recorren con `hasNext()`, y cierran con `endArray()`.
  - Objetos: Consumen `beginObject()`, asignan valores, y cierran con `endObject()`.

```
public List<Poema> readArrayPoemas(JsonReader reader) thro
    // Guardar la lista de poemas del JSON
    List<Poema> poemas = new ArrayList<>();

    reader.beginArray(); // Leemos el  [
    while (reader.hasNext()) { // para cada elemento de array de
        poemas.add(readPoema(reader));
    }
    reader.endArray(); // Leemos el ]
    return poemas;
}

public Poema readPoema(JsonReader reader) throws IOExcep
    // Código de lectura de un poema
}
```

- **Manejo de Tokens Desconocidos**:
  - `skipValue()` se usa para omitir valores desconocidos y evitar conflictos de estructura.
  - `peek()` devuelve el tipo de dato de un elemento JSON

- `nextNull()` Se utiliza para consumir literales nulos (También se puede usar `skipValue()`)