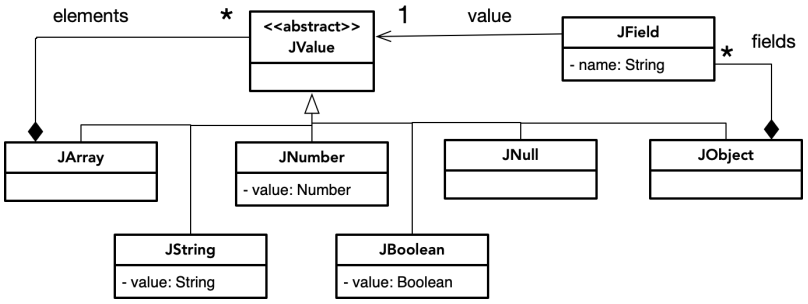
Engenharia de Linguagens de Programação

Projeto JQL (2023/2024)



Neste projeto pretende-se desenvolver uma pequena linguagem de scripting para recolher informação de ficheiros JSON. Os scripts serão capazes de carregar conteúdo JSON de ficheiros, o qual pode ser interrogado para formar novo conteúdo JSON. A técnica de implementação da linguagem pode ser baseada em interpretação ou geração de bytecode.

Exemplo de JSON

O ficheiro JSON seguinte será utilizado para descrever as primitivas prentedidas para a linguagem. O mesmo contém um objeto JSON com dois campos, sendo que o segundo contém um vetor JSON com outros objetos. Atenção que os ficheiros podem conter qualquer valor JSON (pe. vetores JSON).

lei.json

```
{
   "curso":"LEI",
   "ucs":[
      {
         "sigla":"IP",
         "creditos":6,
         "horas":4.5
      },
      {
         "sigla":"POO",
         "creditos":6,
         "horas":4.5
      },
      {
         "sigla":"ELP",
         "creditos":6,
         "horas":3.0
      }
   ]
}
```

Sintaxe

O seguinte exemplo ilustra a linguagem pretendida. Não é necessário que a sintaxe a desenvolver seja exatamente igual, desde que o tipo de funcionalidade seja equivalente.

resumo.jql

```
load $1 to doc                          # carrega json do primeiro argumento
curso = doc.curso                       # "LEI"
ucs = doc.ucs                           # [ {sigla: "IP"...}, {...}, {...}]
siglas = doc.ucs*.sigla                 # ["IP","POO","ELP"]
creditos = doc.ucs*.creditos | sum      # 24
maxHorasUc = doc.ucs*.horas | max       # 4.5
total = ucs | count                     # 3

resumo = {                              # constroi objecto json utilizando
      "curso": curso,                   # variaveis existentes
      "ucs": siglas,
      "creditos": creditos,
      "maxHoras": maxHorasUc,
      "totalUcs": total
}

save resumo to $2                       # grava json no segundo argumento
```

Ao executar o script acima na linha de comandos, teriamos como resultado um novo ficheiro com o conteúdo em baixo.

```
java JQL resumo.jql lei.json lei-resumo.json
```

lei-resumo.json

```
{
   "curso":"LEI",
   "ucs":[
      "IP",
      "POO",
      "ELP"
   ],
   "creditos":24,
   "maxHoras":6.0,
   "totalUcs":3
}
```

Argumentos

Tal como ilustrado, o script poderá ter parâmetros, referidos por $1, $2, etc. Os valores passados na execução serão convertidos para JSON simples (i.e., números, string, ou booleano) e constituem os argumentos do script.

Validação: a utilizanção de um número de argumentos diferente do número de parâmetros deverá dar origem a um erro de execução.

Atribuição

Exceptuando as instruções de carregamento (load) e gravação (save), só existe um tipo de instrução: a atribuição (*var* = <*valor json*>). As variáveis podem ser reatribuídas, mas os valores associados são imutáveis. A parte direita é sempre algo que resulta num valor JSON, quer direto, quer resultante da avaliação de uma expressão.

Numa atribução podemos criar novos valores JSON (pe. em *resumo*) utilizando a sintaxe do próprio JSON, porém podemos referir variáveis anteriores para obter valores.

Validação: a referência a uma variável não existente deverá dar origem a um erro de compilação.

Expressões

As expressões podem ser utilizadas na parte direita das atribuições, e resultam sempre num valor JSON. Através do ponto "." podemos aceder aos campos dos objetos.

O operador asterísco "*" permite exprimir "para cada objeto". No exemplo temos:

```
siglas = doc.ucs*.sigla
```

que deve ser interpretado como "para cada objeto em *doc.ucs*, é recolhido o valor do campo *sigla*", resultando num vetor JSON.

Validação: a referência a um campo não existente num objeto JSON (ou outro valor que não seja objeto) deverá dar origem a um erro de execução.

Operações agregadoras

As operações agregadoras (max / min / count / sum / avg) permitem reduzir um vetor JSON a um valor JSON numérico.

Validação: a utilização de uma operação agregadora em valores inapropriados (pe. *sum* num vetor de booleanos) deverá dar origem a um erro de execução.