Linguagem Para Construção de Questionários

Compiladores 2019/20

Diogo Filipe Amaral Carvalho – 92969
Luís Miguel Dias Santos Pereira Costa - 85044
Maria Inês Seabra Rocha - 93320
Mariana Sousa Pinho Santos - 93257
Rafael Ferreira Baptista - 93367

Índice

Índice de Imagens

Introdução

No decorrer da realização do projeto de grupo Unidade Curricular de Compiladores, o nosso grupo propôs-se a desenvolver uma linguagem capaz de criar e manipular questionários.

A linguagem desenvolvida permite criar questionários capazes de suportar questões de vários tipos diferentes, tais como escolhas múltiplas, verdadeiras e falsas, respostas curtas e longas. Para além disso, é também possível criar configurações relativas à apresentação do questionário propriamente dito, tais como escolher a ordem das perguntas ou definir o tempo máximo da sua duração. Por fim, a linguagem oferece também a possibilidade de o programador criar as suas próprias questões ou então fazer *upload* de um ficheiro com perguntas já feitas, podendo também combinar os dois.

De forma a alcançar as funcionalidades pretendidas para a linguagem, foi necessária a criação de duas gramáticas. A gramática principal, correspondente à linguagem a ser compilada, foi desenvolvida de modo a permitir a implementação das funcionalidades principais da linguagem, isto é, todas as operações necessárias para a construção do questionário em si. A gramática secundária foi desenvolvida com o intuito de permitir a inserção de perguntas numa base de dados, de modo a oferecer ao utilizador a possibilidade de escolher perguntas anteriormente definidas.

De modo a oferecer ao utilizador um maior controlo sobre a apresentação do questionário, e enriquecer um pouco mais a linguagem, adicionámos o suporte para funcionalidades tradicionais das linguagens de programação, tais como estruturas condicionais, ciclos e a importação (sendo que no nosso caso são importadas perguntas e não pacotes). Devido à natureza da nossa linguagem, a impressão na consola e os inputs feitos pelo utilizador são suportados por defeito.

Gramática BaseDados

Sintaxe

A gramática **BaseDados**, definida no ficheiro *BaseDados.g4* permite a declaração de perguntas, de modo serem guardadas como se estivessem numa base de dados, com o objetivo de estarem disponíveis posteriormente. A declaração de uma questão segue a seguinte sintaxe:

```
declaracao: 'questao' ':' TIPO (DIFICULDADE)? pontuacao? '['tema']' STRING? respostas* ;
```

Figura 1: Sintaxe da declaração de uma questão

Exemplos de declaração de perguntas:

```
questao:EscolhaMultipla medio pontuacao=20 [ARTES->DANCA->Q3] " Quais dos seguintes são nomes de passos básicos de Hip Hop?"
inicio
  "Hip Walk" false pontuacao=-20;
  "Criss Cross" true pontuacao=30;
  "Jumping Mouse" false pontuacao=-20;
  "Running Man" true pontuacao=30;
  "Slide Simpson" false pontuacao=-10;
  "Spongebob" true pontuacao=40;
fim
```

Figura 2: Declaração de uma pergunta de escolha múltipla

```
questao:CurtaTextual medio pontuacao=(10,-2) [ARTES->PINTURA->Q3] "Em que país nasceu Pablo Picasso? "
inicio
  "Espanha" pontuacao=100;
fim
```

Figura 3: Declaração de uma pergunta de resposta curta

Gramática completa:

```
grammar BaseDados ;

program: stat EOF;

stat: declaracao* ;

declaracao: 'questao' ':' TIPO (DIFICULDADE)? pontuacao? '['tema']' STRING? respostas;

respostas: 'inicio'? (resp ';')* 'fim'? ;

resp:   STRING BOOLEANO? pontuacao?;

pontuacao: 'pontuacao' '=' valor;

valor:       INT                #IntegerCtx
       | '(' INT ',' INT ')'    #ValorTuploCtx
       ;

tema:
      (TEXTO|TIPO) ('->' tema)    #TemaPlus
    |  (TEXTO|TIPO)               #TemaFinish
    ;


DIFICULDADE: 'facil' | 'medio' | 'dificil' ;
BOOLEANO: 'true' | 'false';
INT: '-'?[0-9]+;

TIPO: [A-Z][a-zA-Z0-9_]+;
TEXTO: [a-zA-Z0-9_]+;
STRING: '"'  .*? '"';
WS: [ \t\n\r]+ -> skip ;
COMMENTS: '#' .*? '\n' '\r'? -> skip;
```

Figura 4: Gramática BaseDados.g4

Execute.java

A classe **Execute** é uma classe que estende **BaseDadosBaseVisitor**. Esta classe é constituída por métodos que dão *override* aos da classe que estende, de modo a realizar operações quando é efetuada uma visita a um elemento da gramática. Esta classe visita a ação de declaração de uma pergunta e todas as outras ações das quais a declaração depende.

De modo a armazenar as questões declaradas, esta classe gera um novo objeto cujo tipo corresponda ao tipo de questão de declarada. As classes que definem estes objetos são subclasses da classe **Questão**, definida no pacote da gramática principal.

Todos as questões declaradas com esta linguagem são guardadas numa lista de modo a ser possível consultá-las através da gramática principal.

```
switch(tipo){
    case "EscolhaMultipla":
        EscolhaMultiplaQuestao em1  = new EscolhaMultiplaQuestao();
        em1.tema = t;
        em1.dificuldade = d;
        em1.pergunta = pergunta;
        em1.pontuacao = pont;
        em1.respostas = respostas;
        //em1.id = id;
        Auxiliar.importQuestoes.add(em1);
        break;
    case "VerdadeiroFalso":
        VerdadeiroFalsoQuestao vf1  = new VerdadeiroFalsoQuestao();
        vf1.tema = t;
        vf1.dificuldade = d;
        vf1.pontuacao = pont;
        vf1.respostas = respostas;
        vf1.pergunta = pergunta;
        //vf1.id = id;
        Auxiliar.importQuestoes.add(vf1);
        break;
    case "LongaTextual":
        LongaTextualQuestao lt1 = new LongaTextualQuestao();
        lt1.tema = t;
        lt1.pergunta = pergunta;
        lt1.dificuldade = d;
        //lt1.id = id;
        Auxiliar.importQuestoes.add(lt1);
        break;
    case "CurtaTextual":
        CurtaTextualQuestao ct1 = new CurtaTextualQuestao();
        ct1.tema = t;
        ct1.pergunta = pergunta;
        ct1.dificuldade = d;
        ct1.pontuacao = pont;
        //ct1.id = id;
        ct1.respostas = respostas;
        Auxiliar.importQuestoes.add(ct1);
        break;
}
```

Figura 5: Criação de uma nova pergunta

Gramática Questionarios

A gramática **Questionarios**, presente no ficheiro *Questionarios.g4* é, tal como o nome indica, a gramática responsável pela construção do questionário propriamente dito**.** Esta gramática pode ser utilizada de forma independente ou, caso o utilizador o pretenda, ser utilizada em conjunta com a gramática **BaseDados**. Caso se escolha a segunda opção, é possível realizar a importação de perguntas declaradas com recurso à gramática **BaseDados**, passando-se assim a ter acesso a essas perguntas.

É de referir que a nossa linguagem trata cada questão como sendo um objeto que possui várias propriedades, sendo que um questionário é constituído por uma ou mais perguntas. É ainda possível definir grupos de perguntas.

Operações suportadas

Tal como foi referido anteriormente, a nossa linguagem suporta as operações de declaração de variáveis, atribuição de valores, declaração de estruturas de dados simples (como uma lista), implementação de estruturas de decisão (*se*, *ouse, senao*), implementação de ciclos (ciclo *for,* na nossa linguagem denominado como *para*), impressão na consola, efetuar comparações e importação de perguntas.

Para além das operações referidas acima, a gramática suporta os seguintes métodos nativos:

- **adicionarquestao()** – permite adicionar uma questão previamente definida a um questionário ou grupo;
- **adicionarquestoes()** – permite adicionar um conjunto de questões a um grupo, sendo possível filtrar por tema, dificuldade e tipo;
- **apresentarMenu()** – apresenta o questionário em si correspondente a um grupo de perguntas, ou questionário, sendo possível especificar o tempo máximo em que deve ser respondido;
- **apresentar()** – permite ao utilizador imprimir na consola;
- **importar()** – permite ao utilizador importar perguntas de um ficheiro;
- **guardarrespostas()** – permite guardar as respostas dadas num determinado grupo ou questionário num ficheiro a definir pelo utilizador.

Declaração de variáveis

A nível de declaração de variáveis, a nossa linguagem permite a declaração de questionários, perguntas, grupos de perguntas, listas, temas, dificuldades e respostas (para além de outros tipos essenciais, como valores booleanos, *strings* de caracteres, números inteiros e reais). Tal como referido anteriormente, as questões são tratadas como um objeto e, como tal, é necessário atribuir-lhes um tipo aquando da sua declaração, tal como acontece nas respostas.

É de salientar que aquando a declaração de uma questão, todas as suas propriedades são automaticamente declaradas, cabendo depois ao utilizador atribuir-

lhes os valores pretendidos. Nem todas estas propriedades são de cariz obrigatório, não sendo assim, por exemplo, necessária a atribuição de um valor.

A declaração de uma variável é feita do seguinte modo:

```
q1 as Questionário.              # declaração de um questionário
tx1 as Questão:CurtaTextual.         # declaração de uma pergunta de
                                     resposta curta
tx2 as Questão:VerdadeiroFalso.      # declaração de uma pergunta de
                                     verdadeiro e falso
```

Figura 6: Declaração de variáveis

A nível de análise semântica, o interpretador começa por analisar se a variável que pretendemos declarar já se encontrava previamente declarada. Caso esteja, é lançado um erro de modo a avisar o utilizador, e a impedir a compilação do código. Caso a variável ainda não esteja declarada, esta é adicionada à tabela de símbolos.

Atribuição de valores a variáveis

A linguagem criada permite a atribuição de valores a variáveis previamente definidas. Tal como referido previamente, para alguns tipos (nomeadamente, questões, grupos e questionários) é atribuído um tipo às variáveis aquando a sua declaração, como tal, apenas é possível atribuir valores às variáveis que sejam do tipo que estas suportem.

Devido à forma como a questões são tratadas, a sintaxe de atribuição de valores a variáveis é ligeiramente diferente quando estas são propriedades de uma questão. Neste caso, é necessário identificar a questão a que propriedade pertence.

```
tx1->tema := [TEMA_A].               # atribuição de valores a
tx1->pergunta := "Questao...?".      propriedades de uma questão
tx1->dificuldade := FACIL.
tx1->pontuacao := 10.
```

Figura 7: Atribuição de valores a propriedades de uma questão

```
lr as Lista<Resposta:EscolhaMultipla>.       # atribuição de
lr := {"Nee.", 100}.                         valores a uma variável
```

Figura 8: Atribuição de valores a uma variável normal

De modo a evitar erros, é necessária fazer uma verificação semântica. Deste modo, o nosso analisador semântico corre uma série de verificações quando visita a atribuição de valores a variáveis com o intuito de detetar os seguintes erros:

- Atribuição de valores a uma variável não declarada;
- Atribuição de uma lista a uma variável do tipo lista que já contenha um valor;
- Atribuição de valores incompatíveis a uma variável.

Caso o analisador semântico não encontre nenhum erro, o valor da variável é atualizado na tabela de símbolos para o valor atribuído pelo utilizador.

Ciclos

A nossa linguagem suporta a definição de ciclos, nomeadamente o ciclo *for,* com uma estrutura e sintaxe semelhante à do ciclo *for-each* em *Java*. Deste modo, quando se define um ciclo for é necessário declarar sobre o que é que vamos iterar, sendo possível iterar sobre listas de qualquer tipo de dados. Por exemplo, é possível iterar sobre listas de perguntas presentes em questionários e grupos de perguntas, visível no exemplo seguinte (que combina ciclos com condições).

```
para (q as Questao em q1->questoes)
     inicio
          t as String.
          t := "LongaTextual".
          se (q->tipo == t)
          inicio
               apresentar("Respostas: " & str(q->respostadada)).
          fim
     fim
```

Figura 9: Iteração sobre as questões de um questionário

De modo a evitar erros, o analisador semântico executa uma série de verificações para garantir a integridade do programa. Caso algum dos argumentos passados na inicialização de um ciclo não se encontre em conformidade com o esperado o analisador indicará um dos seguintes erros:

- Tentativa de iteração sobre uma variável não declarada;

- Tentativa de iterar sobre um objeto que não é uma lista;
- Tentativa de iterar sobre elementos não presentes na lista passada como argumento.

Estruturas de decisão

A nível de estruturas condicionais, a nossa linguagem suporta a utilização das instruções *if*, *else if* e *else*, sob os nomes de **se**, **ouse** e **senao**. Estas estruturas são normalmente utilizadas em conjunto com estruturas de comparação, que serão abordadas no próximo tópico.

Sempre que o analisador semântico visita uma instrução decisão ele apenas verifica se ela se encontra numa cadeia de decisão ou se é uma decisão *standalone*.

```
se (g1->pontuacao < p)          # p é uma variável do tipo inteiro
     inicio                          previamente definida
          apresentar("Não pode prosseguir! Teve: " &
                         str(g1->pontuação) & " pontos")
     fim
senao
     inicio
          apresentar("Parabéns, concluiu a primeira fase").
     fim
```

Figura 10: Implementação de uma estrutura de decisão

Estruturas de comparação

Tal como referido acima, é possível efetuar comparações com a nossa linguagem. As operações suportadas são as seguintes:

- <- menor que
- >- maior que
- <= menor ou igual que
- >= maior ou igual que
- != diferente de
- e
- ou

De modo a prevenir erros, sempre que o analisador semântico visita uma destas ações, ele verifica se os tipos das variáveis a ser comparadas são compatíveis. Caso não o sejam, é apresentada uma mensagem de erro ao utilizador. Também é necessário

que os valores utilizados nestas comparações estejam declarados e passados em forma de variáveis.

Métodos Nativos

De modo a facilitar a vida ao utilizador a nossa linguagem oferece alguns métodos nativos que permitem a realização de algumas operações relativas à criação de questionários.

baralhar

O método *baralhar* permite ao utilizador baralhar a ordem das respostas a serem apresentadas para uma determinada questão.

```
em1->respostas := lr.
em1->respostas%baralhar().
```

Figura 11: Método 'baralhar'

A nível de analise semântica são efetuadas um conjunto de verificações e caso exista algum problema o utilizador recebe um dos seguintes erros:

- Tentativa de utilização do método sem indicar a variável alvo;
- A variável passada como argumento não está declarada;
- A variável passada não é uma Lista.

adicionarquestao

O método *adicionarquestao* permite ao utilizador adicionar uma questão previamente definida a um questionário.

```
q1%adicionarquestao(em1).
q1%adicionarquestao(vf1).
q1%adicionarquestao(tx1).
```

Figura 12: Método 'adicionarquestao'

A nível de analise semântica são efetuadas um conjunto de verificações e caso exista algum problema o utilizador recebe um dos seguintes erros:

- Tentativa de utilização do método sem indicar a variável alvo;
- A variável passada como argumento não está declarada;
- A variável passada não é um Questionário.

adicionarquestoes

Este método é muito semelhante ao anterior, sendo que as questões são adicionadas a um grupo ao invés de um questionário. Para além disso, este método pressupõe que lhe é passado um conjunto de questões ao invés de só uma. Por fim, este método também permite ao utilizador filtrar as questões com base no tema, dificuldade e tipo de pergunta.

```
g2 as Grupo.

q2%adicionarquestoes(q1).
```

Figura 13: Método 'adicionarquestoes'

A nível de analise semântica são efetuadas um conjunto de verificações e caso exista algum problema o utilizador recebe um dos seguintes erros:

- Tentativa de utilização do método sem indicar a variável alvo;
- A variável passada como argumento não está declarada;
- A variável passada não é um Questionário.

importar

O método *importar* permite que o utilizador possa importar perguntas armazenadas num ficheiro. Este método recebe como argumento o ficheiro de origem e a variável de destino para as perguntas.

```
q1%importar("Geografia.txt").
```

Figura 14: Método 'importar'

A nível de análise semântica são efetuadas um conjunto de verificações e caso exista algum problema o utilizador recebe um dos seguintes erros:

- Tentativa de utilização do método sem indicar a variável alvo;
- A variável passada como argumento não está declarada;
- O ficheiro de origem passado não contém um Questionário ou um Grupo.

apresentarMenu

Este método é utilizado para fazer a apresentação do menu correspondente a um grupo de questões. Como argumentos pode aceitar apenas uma variável do tipo Grupo ou Questionário, e pode também aceitar um inteiro que representa o número de segundos em que será possível responder às perguntas apresentadas. Depois de esse tempo (caso tenha sido especificado), as respostas não serão guardadas.

```
apresentarMenu(g2).
```

Figura 15: Método 'apresentarMenu'

A nível de análise semântica é verificado se a variável passada como argumento suporta este método. Caso não suporte é apresentado um erro ao utilizador.

apresentar

O método *apresentar* permite ao utilizador imprimir na consola. Este método suporta concatenação.

```
apresentar("Isto é um print").
```

Figura 16: Método 'apresentar'

A nível de análise semântica, é verificado se a variável passada como argumento suporta este método. Caso não suporte é apresentado um erro ao utilizador.

str

O método *str* permite que o utilizador converta uma variável do tipo Inteiro ou Real para uma variável do tipo *String*. Isto é algo que dá bastante jeito principalmente quando se quer fazer uma concatenação para imprimir na consola.

```
apresentar("Pontos" & str(g1->pontuação)).
```

Figura 17: Exemplo de utilização do método 'str'

A nível de analise semântica, é verificado o número de argumentos passados no método e o tipo da variável passada como argumento. Caso não esteja em conformidade com o esperado é apresentado um dos seguintes erros:

- O número de argumentos passados é diferente do esperado;
- O tipo da variável passada é diferente do esperado (Inteiro ou Real).

Compilador

De modo a executar o processo de compilação da nossa linguagem é necessário correr a classe *QuestionariosMain*, passando-lhe como argumento o ficheiro a ser compilado. O nosso compilador tem como linguagem destino o Java.

Quando o utilizador corre a classe *QuestionariosMain* com o intuito de compilar um programa escrito na nossa linguagem, esta classe irá instanciar um analisador semântico de modo a verificar a existência de erros no ficheiro. À medida que o analisador vai visitando o programa escrito pelo utilizador, as variáveis declaradas irão ser colocadas na tabela de símbolos de modo a poderem ser consultadas posteriormente pelo compilador. Caso o programa contenha erros, o utilizador será notificado pelo analisador semântico e a compilação falhará.

Se o programa que vai ser compilado não possuir nenhum erro semântico, a classe *QuestionariosMain* irá invocar o compilador propriamente dito. O compilador irá visitar a árvore do *Visitor* e, com a ajuda do *String Template*, irá gerar o código fonte em java. Após o término do processo de compilação da nossa linguagem, é necessário compilar o código Java gerado. Para tal basta corre o comando **javac <NomeDoFicheiro>.java.**

Por fim, basta apenas correr o ficheiro Java gerado e poderá começar a responder ao questionário desenvolvido.

Programas de exemplo

De modo a testar o funcionamento do nosso compilador, foram desenvolvidos alguns programas de teste, tanto para a linguagem principal como para a secundária. Neste relatório só iremos apresentar um exemplo para cada linguagem, podendo os restantes exemplos ser consultados nas pastas correspondentes a cada linguagem.

```
#Declaracao variaveis Questionario.
q1 as Questionario.


#Importe das perguntas através de um método import da classe "Questionário".
q1%importar("./exemplos_base_dados/Geografia.txt").

#Indicação do número de perguntas a apresentar.
g as Grupo.
g->nrperguntasaapresentar := 3.

#Formação de um grupo de perguntas com número de perguntas igual ao defenido na variavel "q->numeroperguntas".
q1->questoes % baralhar() .
g % adicionarquestoes(q1).

#Apresentação do Menu de Perguntas e Respostas ao utilizador.
apresentarMenu(g, 20).

#Apresentação da pontuação final.
apresentar("A sua pontuaçao final foi de :" & str(g->pontuacao)).

guardarrespostas(g, "ficheiro.txt"). # guarda todas as respostas/questoes
                                     # no ficheiro passado (faz append)

c as Real.
c := 0.
para (r as Questao em g->questoes)
inicio
    apresentar ("pontuacao total: " & str(r->pontuacao) &  "; pont. obtida: " & str(r->respostadada->pontuacao) ).
    c := c + r->pontuacao * r->respostadada->pontuacao/100 .


fim

apresentar("pontuacao calculada: " & str(c)).

aa as Real.
bb as Real.
aa := 5.
bb := 2.

cc as Real.
cc := aa // bb. # símbolo do resto! (//)

apresentar(str(cc)).
```

Figura 18: Programa de exemplo da linguagem principal

```
questao:EscolhaMultipla facil pontuacao=10 [GEOGRAFIA->PORTUGAL->Q1] "Onde se localiza o aeroporto Francisco Sá Carneiro?"
inicio
  "Faro" false pontuacao=-20 ;
  "Porto" true pontuacao=100 ;
  "Lisboa" false pontuacao=0;
  "Montijo" false pontuacao=-20;
fim

questao:VerdadeiroFalso medio pontuacao=40 [GEOGRAFIA->PORTUGAL->Q2] "Questões de Verdadeiro ou Falso..."
inicio
  "O rio Douro situa-se na Região Norte de Portugal" true pontuacao=(10,-5);
  "A Serra da Estrela é a montanha mais alta de Portugal" false pontuacao=(20,-20);
  "Sertã é uma localidade situada no conselho de Santarém" false pontuacao=(40,-10);
  "Sesimbra é uma vila Portuguesa do Distrito de Setúbal" true pontuacao=(20,-20);
  "O rio Sado situa-se no sul de Portugal e faz fronteira com a Espanha" false pontuacao=(10,-5);
fim

questao:EscolhaMultipla medio pontuacao=20 [GEOGRAFIA->PORTUGAL->Q3] "Em que ilha dos açores podemos ver o vulcão Pico Alto?"
inicio
  "São Miguel" false pontuacao=-20;
  "Corvo" false pontuacao=-20;
  "Santa Maria" false pontuacao=-30;
  "Terceira" true pontuacao=100;
fim
```

Figura 19: Exemplo de programa escrito com a linguagem secundária

Autoavaliação

Diogo – 24,5%

Luís – 17%

Maria - 17%

Mariana - 24,5%

Rafael - 17%