Relatório – Trabalho prático

Programação III

PROLOG – CÓDIGOS AMBIGUOS



**Trabalho realizado por:**

Pedro Grilo, 43012

Diogo Castanho, 42496

1.Introdução

Foi-nos proposta a realização de um trabalho que consistia na implementação de um programa que iria definir Sistemas-L, gerar palavras a partir desses mesmos sistemas, e por último, iria visualmente mostrar-nos essas palavras num gráfico.

Uma Gramática é um sistema formal usado, por exemplo, para definir linguagens de programação. Esta define um conjunto de “letras”, uma “palavra inicial” e um conjunto de “regras de re-escrita”.

Um Sistema-L é um tipo simplificado de gramática com uma interpretação gráfica especialmente adequada a Turtle Graphics.

Quando este processo executado pela gramática Sistema-L se repete e se gera uma palavra nova a partir da anterior, diz-se que ocorre uma iteração, e esta mesma, produz uma linguagem.

As Turtle Graphics são uma variante simples dos caminhos usados nos sistemas gráficos 2D: uma “tartaruga” é controlada por instruções específicas e o seu rasto depois de variadas instruções define um desenho.

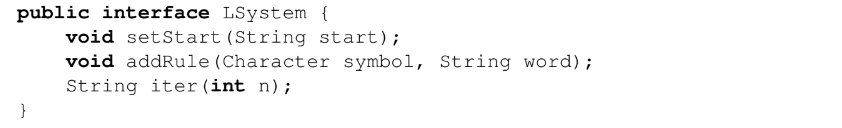
A relação entre os Sistemas-L e os Turtle Graphics é feita pela tradução gráfica das letras do sistema, ou seja, a cada letra de um Sistema-L faz-se corresponder a uma instrução da tartaruga. No fim da palavra teremos então o desenho realizado por esta.

Para a sua resolução seguimos os seguintes passos: em primeiro o código é gerado por um Sistema-L, em segundo compilado para uma representação média e por final é interpretado para produzir uma imagem no ecrã.

1.2 – Bases para o trabalho

Para a resolução do trabalho seguimos os seguintes passos: em primeiro o código é gerado por um Sistema-L, em segundo compilado para uma representação média e por final é interpretado para produzir uma imagem no ecrã.

Para gerar palavras usando o Sistema-L é necessário considerar-mos a palavra inicial, as regras do mesmo e também o número de iterações pretendido.

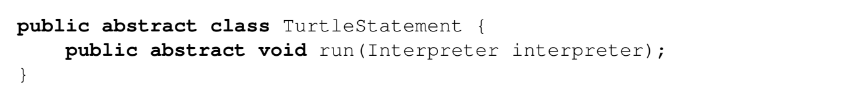
Então fizemos uma interface Sistema-L para gerar palavras desses sistemas, seguindo este exemplo:

Onde o método setStart(String start) é usado para definir a palavra inicial do sistema, e o método addRule(Character symbol, String word) tem como função adicionar a regra depois de termos a palavra inicial.

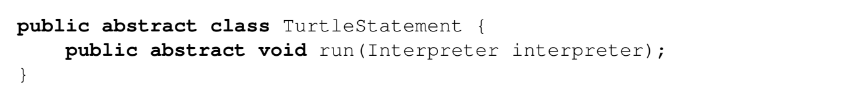
Já iter(n) serve para gerar a palavra que resulta da iteração do sistema n vezes.

Na fase de compilar a palavra tivemos antes que ver como as instruções que provém da tartaruga.

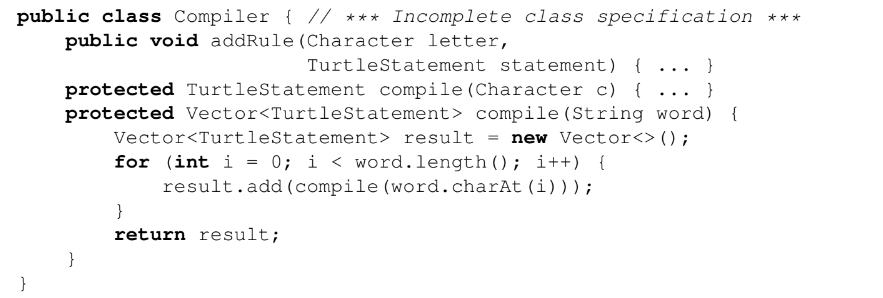
A representação das instruções do Programa Tartaruga assenta numa classe abstrata, Turtle Statement.



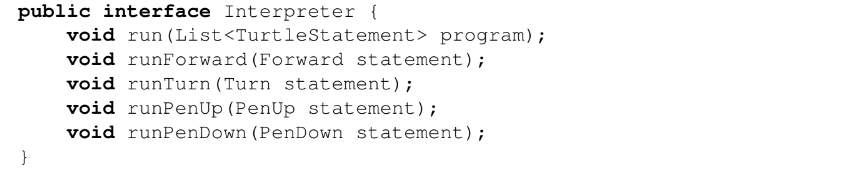
Para representar tipos concretos de instruções estende-se esta classe da seguinte maneira:



Após isto, vimos como compilar palavras para tartaruga da seguinte forma, sendo esta a classe parcial que tivemos como base:



Na fase de interpretação, seguimos também a seguinte interface interpreter:



Resumindo, a interpretação de programas Tartaruga está organizada pelo interface Interpreter, que impõe um conjunto de métodos que ligam o interpretador aos diferentes tipos de instrução.

É em cada um dos métodos run\*\* que o efeito de cada instrução fica definido.

1.3 Decisões tomadas durante o trabalho

O grupo pensou inicialmente, para chegar ao objetivo final, em criar uma classe que gerasse a palavra (segundo a interação introduzida) de qualquer Sistema-L de nível 0.

Após várias tentativas, debug’s e concentração conseguiu-se fazer o esperado inicialmente para a primeira parte do trabalho e foram obtidos os resultados esperados.

Quanto à segunda parte, não houve muita dificuldade para fazer as classes *Forward*, *Turn, PenUp e PenDown*, porém a classe *Compiler* demorou um pouco a compreender o que era realmente preciso fazer e acabou por se obtar por usar o *TreeMap*.

Por fim, após fazer a interface *Interpreter* e completá-la segundo o que era pedido o resultado não foi o esperado inicialmente devido a problemas de escala mas no final acabámos por conseguir colocar tudo a funcionar.

Após várias tentativas e muito esforço o grupo conseguiu perceber o que se passava e o que estava a falhar na última parte do trabalho.

O trabalho não ficou totalmente completo, uma vez que não se fez as classes e objetivos para de cada ponto para mais valores adicionais. Concluindo, achamos que ainda conseguimos pontos muito importantes do trabalho sendo por isso muito importante no nosso progresso da cadeira de *Programação II*.