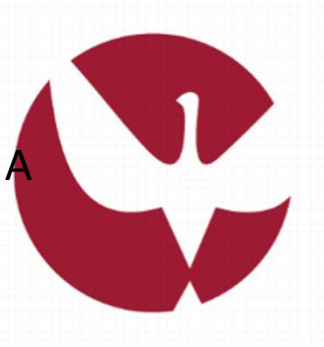
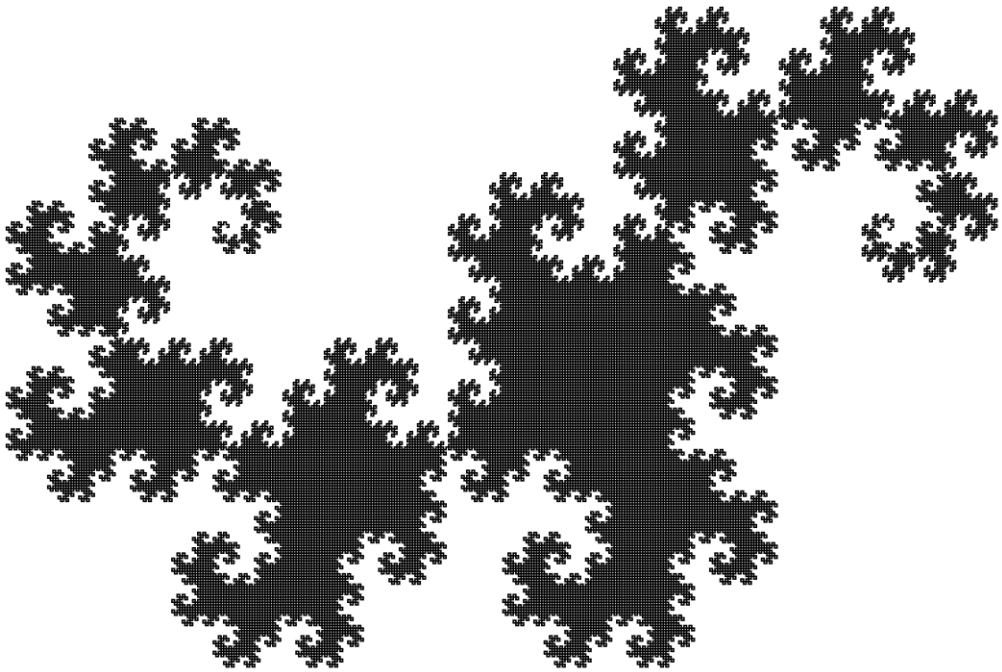
UNIVERSIDADE DE ÉVORA

CURSO DE ENGENHARIA INFORMÁTICA

PROGRAMAÇÃO II 2018/2019



SISTEMAS-L E TURTLE GRAPHICS - RELATÓRIO



Dinis Matos nº 42738

José Santos nº 43017

Introdução

Após uma análise geral do trabalho proposto de Programação II (SISTEMAS-L E TURTLE GRAPHICS), começou-se a colocar os exemplos descritos, no enuciado, no trabalho para a execução de tal. As classes indicadas são KochCurve, TurtleStatement, Compiler, Interpreter, também é indicadaa interface Lsystem e alguns exemplos das classes implementadas. No processo total do trabalho, ao todo, criamos as classes Compiler, Forward, Interface, InterpreterClass, Leap, LSystemClass, Main, PenDown, PenUp, Turn, TurtleStatement e as interfaces Interpreter e LSystem. Todas as classes e interfaces têm a sua própria função.

Classe Compiler

A Classe Compiler cria o **tmap** que é um TreeMap<Character, TurtleStatement>. É ainda constítuida pelas funções **addRule**, em que esta recebe, como argumentos, **letter** (Character) e **statement** (TurtleStatement) e executa tmap.put(letter,statement), pela função **compile**(TurtleStatement), em que esta recebe **c**(Character) e dá **return** a tmap.get(c), e pela função **compile**(Vector<TurtleStatement>), que recebe **word**(String), que cria **result**(Vector<TurtleStatement>) e cria o ciclo for (int i = 0; i < word.length(); i++).

Dentro do ciclo for, é executado result.add(compile(word.charAt(i))) e fora do ciclo mas ainda dentro da função é dado **return** de result.

Esta classe tem como finalidade compilar o programa.

Classe Interface

A classe Interface tem como Objetivo criar L-Systems, novos ou predefinidos, segundo os inputs do utilizador a partir de uma Interface gráfica.

Ao inicio são Inicializadas as variáveis que irão ser usadas nesta classe, o método run() que é chamado na Main, começa por criar as JFrames, JTexfields, JComboBoxs e JLabels e adiciona-as a ArrayLists, para melhor organização, de seguida um addItemListener, para verificar se a opção de L-System é mudada, se sim, um switch-case para alterar as TextFields e JComboBoxs que contêm os valores das regras do compilador e regras para gerar uma palavra.

É chamado o método hideOrShow(...), este método recebe como argumentos componentes do JFrame e serve para os mostrar ou esconder da Frame, consoante a regra do compilador, é também chamado o método reset(...), recebe como argumento todos os JTextFields, JLabels, JComboBoxes, na forma de ArrayList e coloca-os a default, isto é apaga o texto já escrito.

Um addActionListener, que verifica quando é que o Start Button é pressionado, é gerado um L-System tendo em conta os inputs do utilizador

Interface Interpreter

Antes da Interface, são importados o **galapagos.Turtle** e **java.util.***. Na Interface Interpreter existem as funções **run**, que recebe **List<TurtleStatement>**, runForward, que recebe **Forward**, runTurn, que recebe **Turn**, runPenUp, que recebe **PenUp**, runPenDown, que recebe **PenDown**, runLeap, que recebe **Leap**, e por último **runTurtle**. Esta Interface tem como finalidade ligar à classe InterpreterClass.

Classe InterpreterClass

A Classe LSystemClass implementa a Classe LSystem. Dentro desta classe são criadas variáveis **turtle**(Turtle) e **drawing**(TurtleDrawingWindow). Esta classe ainda é constituída pelas funções, **runTurtle**, **run**, **runForward**, **runTurn**, **runPenUp**, **runPenDown** e **runLeap**.

Na função **runForward** é recebido um **Foward** e é executado turtle.move(statement.getDistance()).

Na função **runTurn** é recebido um **Turn** e é executado turtle.turn((statement.getAngle())).

Na função **runPenUp** é recebido um **PenUp** e é executado turtle.penUp().

Na função **runPenDown** é recebido um **PenDown** e é executado turtle.penDown().

Na função **runLeap** é recebido um **Leap** e é executado turtle.penUp(), turtle.move(statement.getDistance()) e turtle.penDown().

A função **runTurtle** apenas cria uma nova turtle e drawing e dá algumas definições.

Interface LSystem

A Interface LSystem tem as funções **setStart**, que recebe uma **String**, e **addRule**, que recebe um **Character** e uma **String** e, por último, **iter** que é recebe um **int**. Esta Interface tem como finalidade ligar à classe LSystemClass.

Classe LSystemClass

A Classe LSystemClass implementa a Classe LSystem. Dentro desta classe são criadas variáveis publicas, String **start**, **texto**, String[] **word** = new String[20], char[] **symbol** = new char[20] e int **cont** = 0. A classe é ainda constituída pelas três funções, **setStart**, **addRule** e **iter**.

A função **setStart** faz que a start da classe seja igual à String recebida.

A função **addRule** faz que o symbol e a word da classe sejam iguais aos Character e String recebidos. Enquanto isso, cont torna-se auxiliar a este processo e é aplicado-lhe uma incrementação.

A função **iter** faz com que texto seja igual a start, depois tem uma condição **while**(n>0), que dentro dele tem a condição **for** (int i=0; i<symbol.length;i++), e que dentro deste é criado uma String **help** que será igual a String.valueOf(symbol[i]). Ainda dentro do ciclo for, texto será igual a texto.replaceAll(help,word[i]). Depois da

condição for, ainda dentro do ciclo while, texto será igual a texto.toUpperCase() e tem um decrementador em n. Já fora do ciclo while, a função terá **return texto**.

A finalidade desta classe é criar a palavra para quaisquer regras e número de iterações.

Class TurtleStatement

Esta é uma classe abstrata que tem somente uma função, **run**, que recebe um **Interpreter**. A finalidade desta classe é interligar as classes Leap, Forward, PenUp, PenDown, Turn.

Classe Leap

Classe Leap é uma classe que extende a Classe TurtleStatement. Cria a variável **distance(double)**, tem uma função **Leap** que recebe um **distance**(double) e depois igual a variável distance dentro da classe a esta nova variável criada. Outra função é **getDistance** que retorna distance. A última função é **run** que recebe um **Interpreter** e executa **interpreter.runLeap(this).** Esta classe é ultilizada para receber e devolver o valor de Leap e executar interpreter.runLeap(this).

Classe Forward

Classe Forward é uma classe que extende a Classe TurtleStatement. Cria a variável **distance(double)**, tem uma função **Forward** que recebe um **distance**(double) e depois igual a variável distance dentro da classe a esta nova variável criada. Outra função é **getDistance** que retorna distance. A última função é **run** que recebe um **Interpreter** e executa **interpreter.runForward(this).** Esta classe é ultilizada para receber e devolver o valor de Forward e executar interpreter.runForward(this).

Classe PenDown

A Classe PenDown extende a Classe TurtleStatement é constituída pelo seu construtor e pela função **run**. Esta função recebe um **Interpreter** e executa **interpreter.runPenDown(this)**. Esta classe é ultilizada para executar interpreter.runPenDown(this).

Classe PenUp

A Classe PenUp extende a Classe TurtleStatement é constituída pelo seu construtor e pela função **run**. Esta função recebe um **Interpreter** e executa **interpreter.runPenUp(this)**. Esta classe é ultilizada para executar interpreter.runPenUp(this).

Classe Turn

Classe Turn é uma classe que extende a Classe TurtleStatement. Cria a variável **angle(double)**, tem uma função **Turn** que recebe um **angle**(double) e depois igual a variável angle dentro da classe a esta nova variável criada. Outra função é **getAngle** que retorna angle. A última função é **run** que recebe um **Interpreter** e executa **interpreter.runTurn(this).** Esta classe é ultilizada para receber e devolver o valor de Turn e executar interpreter.runTurn(this).

Classe Main

Esta é a classe principal pois é a classe que o programa executa. A Classe Main tem dentro dela apenas a função **main** cria uma **Classe Interface** chamada **fun** e executa **fun.run()**.