INSTITUTO SUPERIOR DE ENGENHARIA DE COIMBRA

INSTITUTO POLITÉCNICO DE COIMBRA

JavaLife
Trabalho Prático – Programação Avançada

João Duarte Nº 2020122715
André Dias Nº2021140917
Rúben Agostinho Nº2020157100

COIMBRA

31 de maio de 2024

Índice

Introdução

No âmbito da cadeira de Programação Avançada, pretende-se desenvolver uma aplicação na linguagem Java que simule a evolução de um ecossistema ao longo do tempo. Este ecossistema evolui numa escala de unidades temporais. Neste ecossistema existem três tipos de elementos, dois que reagem ao avanço do tempo: elementos do tipo Fauna e elementos do tipo Flora e um elemento que não reage ao avanço desse tempo, os elementos do tipo Inanimado.

Decisões tomadas

Quando um elemento do tipo Flora está a servir de alimento para um elemento do tipo Fauna, a mesma não ganha força por uma questão de coerência e para evitar aparecimento de bugs.

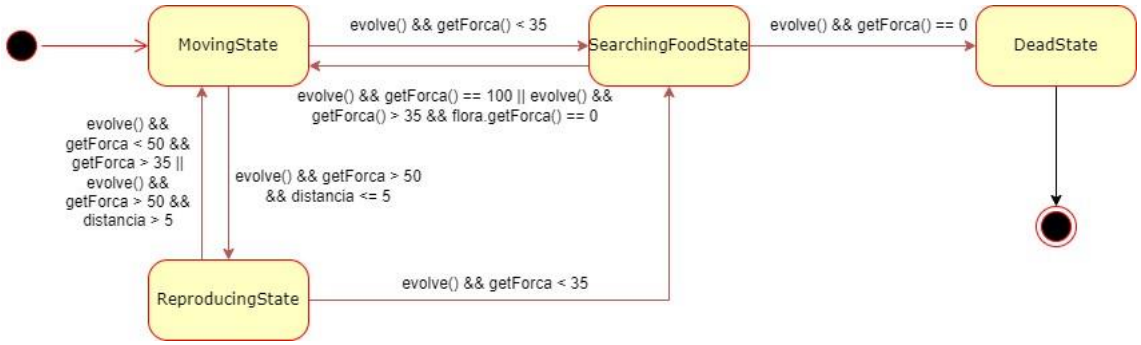
Caso a força do elemento Fauna não esteja abaixo de 35 ou acima de 50, condições que causam mudança de estado, a forma como o elemento se movimenta é feita de forma random sorteando as direções possíveis caso não exista um elemento inanimado para o qual o animal não pode sobrepor.

Para facilitar na gestão das áreas, criámos funções de auxílio na classe Área. Uma função para verificar se as áreas de dois elementos se intersetam, outra função para calcular a distância entre dois pontos e outra para calcular o elemento mais próximo.

Fauna -> classe context e data

Pequeno menu inicial apenas para entrar no simulador e mostrar os créditos de desenvolvimento da aplicação.

Diagrama da máquina de estados



MovingState

Como referido no enunciado, quando cada elemento Fauna é criado encontra-se em movimento. Este é o primeiro estado da aplicação responsável pelo movimento dos elementos do tipo Fauna.

SearchingFoodState

Este estado é desencadeado quando a força do elemento fauna é inferior a 35 e é responsável por mover o elemento do tipo Fauna, na função evolve(), para o elemento do tipo Flora mais próximo com o objetivo de se alimentar. Caso a força do elemento atinja 100 ou seja superior a 35 mas o elemento do tipo Flora acabe por morrer, o elemento do tipo Fauna regressa o estado MovingState.

DeadState

Enquanto está á procura de comida, se o resultado da função getForca() for igual a 0, o elemento do tipo Fauna vai acabar por morrer entrando no estado DeadState.

ReproducingState

Este estado é responsável pela reprodução dos elementos do tipo Fauna, e é desencadeado caso o elemento tenha uma força superior a 50 e esteja a uma

distância inferior a 5 de outro elemento do tipo Fauna. Caso a sua força seja inferior a 50 e maior que 35 o elemento regressa ao movingState, o que acontece também, mesmo que a força seja superior a 50, o elemento esteja a uma distância maior que 5 do outro elemento. Se ao reproduzir-se o elemento do tipo Fauna ficar com uma força inferior a 35 então passa para o estado SerchingFoodState.

Descrição de classes

AbstractCommand

Classe base para todos os comandos que interagem com o Ecossistema. Tem como objetivo disponibilizar as implementações padrão para as operações da interface ICommand.

AddFaunaCommand

Representa a implementação de um comando concreto. Tem como objetivo adicionar um elemento do tipo Fauna. Nesta classe estão definidas as orações de execute(), undo() e redo(), com o objetivo de criar uma nova fauna ou reverter o comando.

AddFloraCommand

Representa a implementação de um comando concreto. Tem como objetivo adicionar um elemento do tipo Flora. Nesta classe estão definidas as orações de execute(), undo() e redo(), com o objetivo de criar uma nova flora ou reverter o comando.

AddInanimadoCommand

Representa a implementação de um comando concreto. Tem como objetivo adicionar um elemento do tipo Inanimado. Nesta classe estão definidas as orações de execute(), undo() e redo(), com o objetivo de criar um novo inanimado ou reverter o comando.

ApplyForcaCommand

Representa a implementação de um comando concreto. Tem como objetivo aplicar força a um elemento do tipo Fauna. Basta selecionar o comando na TopBar e depois clickar no elemento a que deve ser aplicada a força.

ApplyHerbicidaCommand

Representa a implementação de um comando concreto. Tem como objetivo remover um elemento do tipo Flora. Basta selecionar o comando na TopBar e depois clickar no elemento que se quer remover ou então arrastar o rato por vários elementos.

ApplySolCommand

Representa a implementação de um comando concreto. Tem como objetivo reduzir a velocidade de movimentação dos elementos do tipo Fauna e aumentar o ganho de força dos elementos do tipo Flora. Basta selecionar o comando na TopBar para o efeito acontecer.

CommandManager

É o gestor de comandos, responsável por executar os comandos, guardar um histórico dos comandos realizados e guardar os comandos que foram revertidos, podendo voltar a executá-los.

ICommand

Interface base para todos os comandos. Todos os comandos implementados terão uma ação de execute(), undo() e redo().

Area

Classe que define a região com coordenadas específicas com base os lados superior, inferior, esquerdo e direito. Fornece métodos para verificar a interseção

entre duas áreas, calcular distâncias entre dois pontos e encontrar a menor distância entre duas áreas.

Ecossistema

Classe que oferece uma simulação detalhada do ambiente, permitindo aos utilizadores visualizarem as interações entre os elementos do ecossistema. Implementa a interface IGameEngineEvolve de forma a ser capaz de simular as mudanças no ecossistema ao longo do tempo.

Elemento

Define os tipos de elementos que podem estar presentes no ecossistema durante a simulação.

ElementoBase

Fauna

Classe que modela os elementos fauna no ecossistema. Estes elementos representam animais, que possuem características como força, velocidade de movimento e a capacidade de se reproduzirem e alimentarem. Esta classe desempenha também a função de classe "context" no padrão FSM, ou seja, gere as mudanças de estado, permitindo que a fauna altere o seu comportamento conforme o estado atual e os eventos no ambiente. Para além disso, a fauna também atua como classe de dados, guardando as informações sobre cada elemento fauna.

Flora

Classe que modela os elementos do tipo Flora no ecossistema. Estes elementos representam plantas que desempenham o papel de servir de alimento aos elementos do tipo fauna e possuem a capacidade de se reproduzir.

IElemento

Classe que possui todas as operações comuns que são necessárias para a interação com os elementos do ecossistema.

IElementoComForça

Possui todas as operações relacionadas à obtenção e modificação de força dos elementos Fauna e Flora.

IElementoComImagem

Possui todas as operações relacionadas à obtenção e modificação da imagem dos elementos Fauna e Flora.

Inanimado

Classe que modela os elementos do tipo Inanimado no ecossistema. Estes elementos não possuem qualquer comportamento e não são sensíveis ao avanço do tempo, servem principalmente como obstáculos no ecossistema.

IJavaLifeState

Representa uma interface base para todos os estados da aplicação. Tem como objetivo criar todos os métodos a serem implementados pelos diversos estados, sendo estes métodos representativos da transição entre estados e também das operações que podem ser realizadas em cada estado.

DeadState

Representa a implementação de um estado.

MovingState

Representa a implementação de um estado. Tem como objetivo a movimentação de um elemento do tipo Fauna no ambiente e a movimentação em direção ao elemento Fauna mais próximo quando a sua força for superior a 50 com o objetivo de se reproduzir.

ReproducingState

Representa a implementação de um estado. Tem como objetivo a reprodução de um elemento do tipo Fauna caso a sua força o permita e caso a distância se mantenha inferior a 5.

SearchingFoodState

Representa a implementação de um estado. Tem como objetivo o movimento de um elemento do tipo Fauna para um elemento do tipo Flora, caso exista, com o objetivo de se alimentar, caso não exista o elemento do tipo Fauna vai perseguir o elemento Fauna com menos força para o atacar a obter a sua força.

JavaLifeState

Representa uma classe enumerável com todos os estados possíveis. Tem ainda o objetivo de criar instâncias das classes que representam os diversos estados, usando para isso o padrão Factory.

JavaLifeStateAdapter

Representa as implementações por omissão dos métodos definidos na interface IJavaLifeState. Tem como objetivo criar implementações por omissão, desta forma os diversos estados apenas terão de redefinir os métodos que utilizem. Fornece ainda um método que permite alterar o estado atual no contexto.

IMemento

É implementada pela classe Memento e tem como objetivo guardados os dados/estados de uma snapshot.

IOriginator

Contém as funções que devem ser implementas pelo objeto que possui os dados/estado que se pretende salvaguardar.

Memento

Responsável por criar e armazenar snapshots aos dados e estado num preciso momento.

MyOriginator

Responsável por lidar com a criação e restauração de snapshots do estado do ecossistema, permitindo que os utilizadores restaurem o estado anterior do ecossistema.

SimuladorManager

Responsável por gerir a simulação do ecossistema. Fornece as funcionalidades para a interface do utilizador bem como a aplicação dos eventos. Suporta também as funcionalidades de guardar e restaurar o ecossistema.

ImageManager

Classe responsável por gerir imagens na interface gráfica do utilizador.

CreditsUI

Mostra os créditos de desenvolvimento da aplicação.

EcossistemaUI

Parte da interface gráfica de utilizador da simulação. Representa a visualização do ecossistema no ambiente gráfico utilizando JavaFX.

MainMenuUI

Responsável por criar e mostrar o primeiro menu da aplicação que inclui a opção de começar o simulador, ver os créditos e sair.

RootPane

Responsável por organizer a estrutura principal da aplicação da interface gráfica do utilizador.

TopBarMenu

Responsável por criar e gerir o menu superior da interface gráfica.

Relação entre classes

Máquina de estados

A interface IJavaLifeState é implementada pela classe JavaLifeStateAdapter, que por sua vez é estenda pelas classes que representam a máquina de estados: DeadState, MovingState, ReproducingState, SearchingFoodState.

A classe JavaLifeState está responsável por gerir os estados, na medida em que tem uma factory que cria uma instância da classe conforme o estado atual.

A classe context que neste caso é a classe Fauna está responsável por guardar os dados da execução da aplicação, bem como o estado atual.

Comandos

A interface ICommand é implementada pela classe AbstractCommand, que é estendida pelas classes que representam os comandos concretos. Estes comandos são geridos e invocados pela classe CommandManager.

Dados

A classe Ecossistema mantém um conjunto de elementos, que podem ser instâncias da classe Fauna. O Ecossistema usa esses elementos para representar a fauna no ecossistema da simulação.

A classe SimuladorManager gere a simulação do ecossistema e, como tal, interage com elementos da classe Fauna. Por exemplo, o SimuladorManager pode adicionar ou remover elementos de Fauna do Ecossistema.

A classe AddFaunaCommand implementa um comando para adicionar um novo elemento do tipo Fauna. Quando o comando é executado, uma nova Fauna é criada e adicionada ao Ecossistema.

A classe JavaLifeStateAdapter é usada para adaptar o estado de uma Fauna para a interface IJavaLifeState. Isso permite que o estado de uma Fauna seja gerido usando o padrão de estado.

A classe MyOriginator implementa a funcionalidade de memento para guardar e restaurar o estado do Ecossistema. Como parte desse estado, o MyOriginator salva e restaura elementos do tipo Fauna.

Funcionalidades Implementadas

Descrição funcionalidade/ requisito	**Implementado**	**Não Implementado**	**Implementada Parcialmente**
Inanimados	X		
Flora	X		
Fauna	X		
Criar Ficheiro	X		
Abrir gravação	X		
Gravar simulação	X		
Exportar para ficheiro CSV	X		
Importar de ficheiro CSV	X		
Sair da simulação	X		
Configurações gerais do ecossistema	X		
Adicionar elemento Inanimado (Padrão Command)	X		

Adicionar elemento Flora (Padrão Command)	X		
Adicionar elemento Fauna (Padrão Command)	X		
Editar elemento		X	
Eliminar elemento			X(Dá para eliminar a cerca)
Undo/Redo			X(Apenas para os elementos Fauna e Flora)
Configuração da simulação		X	
Executar/Parar		X	
Pausar/Continuar	X		
Gravar snapshot (Memento)	X		
Restaurar snapshot(Memento)	X		
Aplicar Sol	X		
Aplicar Herbicida	X		
Aplicar Força	X		
Máquina de estados	X		
Confirmar se o utilizador deseja gravar antes de sair	X		
GameEngine	X		