Uma imagem com Tipo de letra, texto, Gráficos, design gráfico

Descrição gerada automaticamente

**Departamento de Engenharia Informática e de Sistemas**

**Programação Avançada**

**Trabalho Prático**

***Relatório Implementação JavaLife***

João Miguel Vicente Ventura.: 2022119090

João Pedro Mendes Redondo.: 2022118679

Quévin Duarte Moderno.: 2019135563

31 de maio de 2024

Ano Letivo 2023/2024 | 2º Semestre

**Índice**

[1. Decisões de Implementação 3](#_Toc168172087)

[2. Diagrama da Máquina de Estados 4](#_Toc168172088)

[3. Outros Diagramas de Padrões de Programação 5](#_Toc168172089)

[4. Resumo Funcionalidades 6](#_Toc168172090)

# Decisões de Implementação

**Elemento Fauna**

Devido à evolução temporal ser um processo individual e independente para cada membro da fauna, e não uma evolução simultânea de todo o ecossistema, optou-se por uma abordagem em que cada animal reage e evolui, respetivamente, de acordo com o seu estado atual e as condições do ambiente em seu redor.

Cada elemento pertencente à fauna, no ecossistema, é dotado da sua própria máquina de estados. A razão principal para a tomada dessa decisão prende-se com o facto de a fauna ser o único tipo de elemento que passa por um processo de evolução. Além disso, cada elemento da funa possui um ciclo de vida único e distinto dos outros, o que, só por si, justifica a necessidade de uma evolução individualizada.

**Máquina de Estados**

A máquina de estados é um mecanismo essencial para a orientação do comportamento do elemento fauna. É a mesma que informa e interpreta o momento do ciclo de vida em que o elemento se encontra e determina as ações apropriadas que o animal deve realizar. No entanto, é importante referir e destacar que o próprio animal é responsável por saber e garantir a execução das ações determinadas pela sua própria máquina de estados.

# Diagrama da Máquina de Estados

Uma imagem com texto, diagrama, Esquema, file

Descrição gerada automaticamente

Figura 1. Diagrama da Máquina de Estados

# Outros Diagramas de Padrões de Programação

***Facade Pattern***

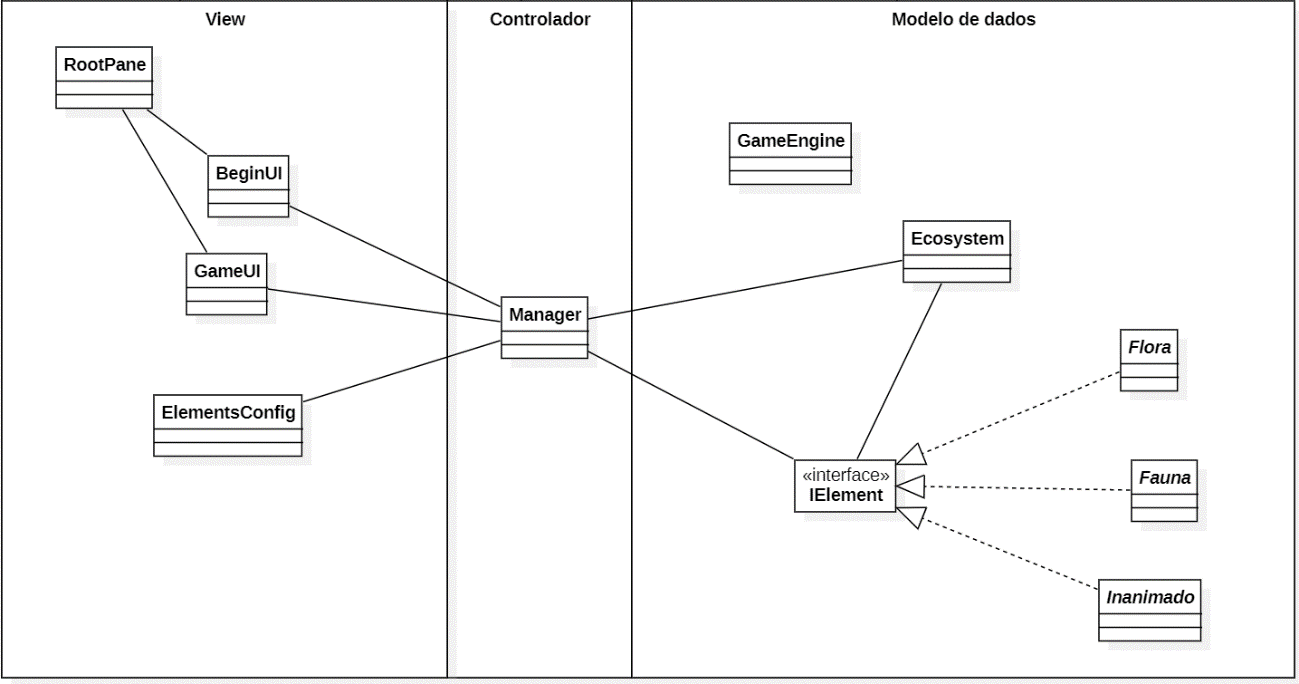


Figura 2. Diagrama do *Facade Pattern*

***Command Pattern***

Uma imagem com diagrama, file, texto, Paralelo

Descrição gerada automaticamente

Figura 3. Diagrama do *Command Pattern*

# Classes

**pt.isec.pa.javalife.gameengine**

**GameEngine**

Representa o motor do jogo, controlando o seu estado, os clientes, a *thread* que executa de forma assíncrona e contínua, e é responsável por executar as iterações do jogo enquanto o seu estado permitir. De forma geral, fornece a estrutura básica que controla o ciclo de vida e de execução do jogo, integrando clientes e notificando-os sobre os eventos importantes. Implementa a interface IGameEngine.

**pt.isec.pa.javalife.model.commands**

**AbstractCommand**

Classe abstrata que serve como abstração genérica para diferentes tipos de comandos. Implementa a interface ICommand.

**CommandAddElement**, **CommandEditElement**, **CommandRemoveElement**, **CommandResize**, **CommandSetTimeStep**

Representam comandos específicos para gerir os elementos do ecossistema, fornecendo diversos métodos para adição, edição, remoção e ajuste do tamanho.

**CommandManager**

Efetua a gestão do histórico de comandos executados, permitindo desfazer e refazer operações.

**pt.isec.pa.javalife.model.data.elements**

**Animal**

Estende a classe fauna, representando um tipo específico de elemento do ecossistema, implementando comportamentos específicos.

**ElementStandard**

Funciona como abstração genérica para os elementos do ecossistema, definindo os atributos e comportamentos comuns compartilhados entre diferentes tipos de elementos.

**Fauna**

Fornece a estrutura básica que representa os animais do ecossistema, com métodos para controlar o seu comportamento, movimento e interações com outros elementos. Dela derivam outras subclasses, como **Animal**.

**Flora**

Representa a estrutura das plantas do ecossistema, controlando o crescimento, reprodução e interação com o ambiente. Dela derivam subclasses concretas que são definidas à posteriori.

**Grass**

Representa um tipo específico de planta, funcionando como uma extensão da classe **Flora**, herdando as suas características e comportamentos básicos.

**Inanimado**

Fornece uma estrutura básica para representar elementos inanimados do ecossistema. Sendo uma classe abstrata selada, define também um conjunto específico de subclasses que podem ser usadas para representar diferentes tipos de elementos inanimados.

**Rock**

Representa uma rocha no ecossistema e é uma subclasse de **Inanimado**, partilhando as suas características básicas. Não apresenta comportamentos específicos pois trata-se de um elemento inanimado.

**pt.isec.pa.javalife.model.data**

**ElementFactory**

Fornece métodos para criar diferentes instâncias no ecossistema, clonar elementos e criar elementos de interface com base nos elementos existentes.

**pt.isec.pa.javalife.model.fsm.states**

**AttackingState**, **EatingState**, **FaunaSearchState**, **FoodSearchState**, **MatingState**, **MovingState**

Gerem os comportamentos da fauna no ecossistema, representando cada estado da máquina de estados, tomando decisões com base no estado atual e nas condições do ambiente no momento.

**pt.isec.pa.javalife.model.fsm**

**ClientContext**

Responsável pela gestão do contexto atual do elemento Fauna. Delega as solicitações para o estado atual e trata da sua própria evolução.

**ClienteStateAdapter**

Classe projetada para funcionar como a classe base e que pode ser estendida por todos os estados. Implementa a interface IClientState.

**Ecossystem**

Responsável pela gestão e manipulação do ecossistema da simulação. Gere os elementos e a sua evolução.

**pt.isec.pa.javalife.model**

**Manager**

Interpreta os pedidos provenientes da interface gráfica e ordena as ações no modelo de dados.

# Resumo Funcionalidades

|  |  |  |
| --- | --- | --- |
| **ID** | **Descrição funcionalidade / requisito** | **Estado** |
|  | Serialização para gravação e leitura de simulação em ficheiros | Implementado |
|  | Padrão Command para a Configuração dos parâmetros do ecossistema e adição, edição e remoção de elementos à simulação com operações de Undo e Redo | Implementado |
|  | Padrão FSM: para a gestão dos estados dos elementos do tipo fauna durante a simulação | Implementado |
|  | Padrão Factory para a criação dos elementos dos diversos tipos; | Implementado |
|  | Padrão Memento para Criação snapshots e Restauro de momentos da simulação anteriormente gravados. | implementado |
|  | Padrão Multiton para a gestão das imagens usadas para cada elemento do tipo fauna. | Implementado |
|  | Configuração da simulação – por exemplo, para definir a unidade de tempo e as dimensões da área visual de simulação | Implementado |
|  | Executar/Parar e Pausar/Continuar; | Implementado |
|  | Eventos: aplicar Sol; aplicar herbicida; injetar força | Implementado |