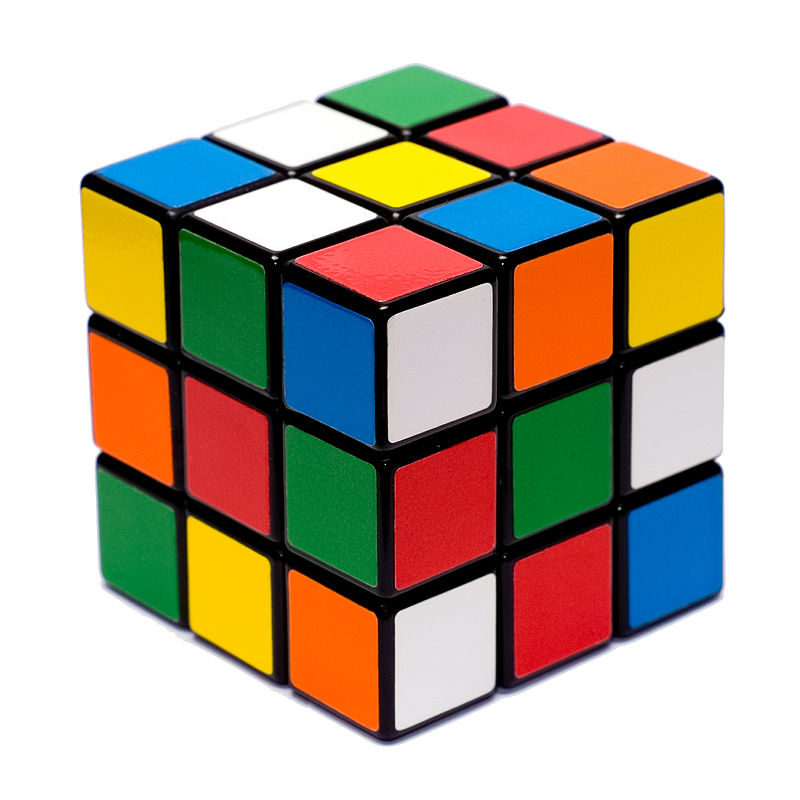
**1. Introdução**

O cubo de Rubik, também conhecido como cubo mágico, é um quebra-cabeça tridimensional no qual utiliza-se uma sequência de movimentos lógicos para poder resolvê-lo. A versão padrão do cubo (3x3), mostrada pela **Figura 1**, é composta por 26 subcubos, dos quais 6 compõe o centro, 12 compõe as bordas e 8 compõe as quinas. Cada parte visível do cubo é rotulada por uma cor e cada movimento altera a posição de quatro quinas e quatro bordas. Assim, a meta é realizar pequenos movimentos no cubo, a partir de um estado inicial, a fim de agrupar todos os rótulos de mesma cor em uma mesma face. Porém, resolver o cubo mágico não é trivial, uma vez que, exitem 4, 3 × 10¹⁹ estados diferentes que podem ser atingidos a partir de qualquer configuração dada.

**Figura 1:** Cubo mágico padrão

O objetivo desse trabalho é utilizar o algoritmo genético para resolver o cubo, a partir de um dado estado. Com isso, busca-se uma solução que aproxime-se da solução ótima por meio da evolução de m população, nas quais estão representadas possíveis soluções do espaço de busca do problema.

A seção 2 deste trabalho discute a modelagem do cubo mágico para um algoritmo genética, destacando pontos como: a representação dos indivíduos, a representação dos operadores, dentre outros. A seção 3 exibe as decisões de implementação, bem como a representação das classes e demais estruturas utilizadas para implementar a resolução deste problema. A seção 4 informa algumas instruções para execução do algoritmo implementado. A seção 5 apresenta os experimentos realizados e a metodologia adotada. A seção 6 apresenta os resultados e análises dos experimentos. Por fim, a seção 6 conclui este trabalho.

**2. Modelagem do Cubo Mágico**

A modelagem do cubo mágico em uma solução baseada no algoritmo genético foi dividida em três fases: modelagem do indivíduo, escolha de uma função de avaliação (Fitness), definição dos operadores de cruzamento e mutação utilizados para criação de uma nova população, e apresentação do algoritmo proposto. Cada uma dessas fases é apresentada a seguir.

**2.1. Modelagem do Indivíduo**

Em qualquer problema que utiliza algoritmos genéticos como possível solução é necessário realizar a formulação de duas características essenciais desses problema: o genótipo, fator que define a composição das características de um indivíduo, e o fenótipo, parte que permite visualização das características de um indivíduo, definidas pelo genótipo.

Nesse problema, um indivíduo representa uma solução válida para a resolução do cubo, ou seja, uma sequência de movimentos que a partir de um dado estado do cubo, consiga encaixar cada uma das cores em uma respectiva face, resolvendo assim o cubo. Essa sequência de movimentos que compõe um indivíduo constitui o genótipo. Porém, apenas as sequências de movimentos por si só, não permite que um usuário consiga visualizar a resolução do cubo. Assim, é necessário que cada indivíduo possua a configuração inicial do cubo, para que se possa visualizar o resultado da aplicação do genótipo e analisar a qualidade da solução. Essa representação do cubo compõe o fenótipo de um indivíduo.

Um primeiro passo na representação de um indivíduo foi definir quais seriam os movimentos que formariam o seu genótipo. Para isso, foram criados dois tipos de movimentos: os movimentos considerados primários e os movimentos considerados compostos.

Os movimentos primários constituem 18 rotações simples das faces dos cubos, identificados por um caractere, que podem comportar-se de três formas diferentes: 90º sentido horário, 90º sentido anti-horário e 180º sentido horário. Os movimentos estão especificados na **Tabela 1**, junto com sua respectiva ação.

|  |  |
| --- | --- |
| **Movimento** | **Ação** |
| B | Rotaciona a face do fundo do cubo 90º em sentido horário. |
| D | Rotaciona a face de baixo do cubo 90º em sentido horário. |
| F | Rotaciona a face da frente do cubo 90º em sentido horário. |
| L | Rotaciona a face da esquerda do cubo 90º em sentido horário. |
| R | Rotaciona a face da direita do cubo 90º em sentido horário. |
| U | Rotaciona a face de cima do cubo 90º em sentido horário. |
| B1 | Rotaciona a face do fundo do cubo 90º em sentido anti-horário. |
| D1 | Rotaciona a face de baixo do cubo 90º em sentido anti-horário. |
| F1 | Rotaciona a face da frente do cubo 90º em sentido anti-horário. |
| L1 | Rotaciona a face da esquerda do cubo 90º em sentido anti-horário. |
| R1 | Rotaciona a face da direita do cubo 90º em sentido anti-horário. |
| U1 | Rotaciona a face de cima do cubo 90º em sentido anti-horário. |
| B2 | Rotaciona a face do fundo do cubo 180º em sentido horário. |
| D2 | Rotaciona a face de baixo do cubo 180º em sentido horário. |
| F2 | Rotaciona a face da frente do cubo 180º em sentido horário. |
| L2 | Rotaciona a face da esquerda do cubo 180º em sentido horário. |
| R2 | Rotaciona a face da direita do cubo 180º em sentido horário. |
| U2 | Rotaciona a face de cima do cubo 180º em sentido horário. |

**Tabela 1.** Descrição dos movimentos primários e suas respectivas ações.

Os movimentos compostos são sequências de movimentos primários agrupados, os quais ao serem executados realizam pequenas alterações no cubo. Esses movimentos foram propostos por Herdy (1994) e foram incorporados a este trabalho como forma de manter uma semântica na alteração do cubo, uma vez que os movimentos primários, por si só, poderiam deixar as soluções muito aleatórias, e em função disso, gerar soluções ruins. Os movimentos compostos estão especificados na **Tabela 2**.

|  |  |  |
| --- | --- | --- |
| **Movimento** | **Nome do Movimento** | **Sequência de Movimentos Primários Correspondente** |
| TEFCW | Two Edge Flip CW | F R B L U L1 U B1 R1 F1 L1 U1 L U1 |
| TEFCCW | Two Edge Flip CCW | F1 L1 B1 R1 U1 R U1 B L F R U R1 U |
| TCFCW | Two Corner Flip CW | L D1 L1 F1 D1 F U F1 D F L D L1 U1 |
| TCFCCW | Two Corner Flip CCW | R1 D R F D F1 U1 F D1 F1 R1 D1 R U |
| TESCW | Three Edge Swap CW | U F2 U1 R1 D1 L1 F2 L D R |
| TESCCW | Three Edge Swap CCW | U1 F2 U L D R F2 R1 D1 L1 |
| TECSCW | Two Edge/Corner Swap CW | R1 U R U1 R1 U F R B1 R B R F1 R2 |
| TECSCCW | Two Edge/Corner Swap CCW | L U1 L1 U L U1 F1 L1 B L1 B1 L1 F L2 |
| TCSCW | Three Corner Swap CW | F1 U B U1 F U B1 U1 |
| TCSCCW | Three Corner Swap CCW | F U1 B1 U F1 U1 B U |
| TIESCW | Three Inslice Edge Swap CW | R L1 U2 R1 L F2 |
| TIESCCW | Three Inslice Edge Swap CCW | L1 R U2 L R1 F2 |

**Tabela 2.** Descrição dos movimentos compostos.

A partir desses dois grupos de movimentos é possível representar soluções para a resolução do Cubo mágico. Neste trabalho, foi definido um tamanho máximo de 30 genes, ou seja, 30 sequências de movimentos para cada um dos cromossomos existentes na população, que estão representados em forma de um vetor. Além desta representação ser simples, ela facilita a realização das operações de cruzamentos e mutações, cujas estão descritas na seção 2. 3 e 2.4 respectivamente.

O fenótipo referente a cada indivíduo é representado pelo próprio cubo mágico. Quando um novo cromossomo é criado, por padrão é atribuído a configuração inicial do cubo que se deseja encontrar uma solução. Após atribuir os movimentos no genótipo de cada indivíduo, os movimentos são executados, e o fenótipo é atualizado de acordo com cada nova configuração sofrida pelo cubo, em decorrência da aplicação dos movimentos. Ao final desses movimentos, a aptidão deste indivíduo é calculada por meio dessa configuração final do cubo. Para uma maior facilidade na execução dos movimentos, o fenótipo está representado por uma lista de matrizes 3x3, onde cada matriz representa a configuração de uma face existente no cubo, e são armazenadas em uma lista. A **Figura 2** ilustra essa representação do fenótipo.

**Colocar aqui a representação do Fenótipo**

**2.2. Função Fitness**

A função Fitness utilizada neste trabalho é dada pela soma de penalidades atribuídas a alguns aspectos posicionados de forma errada, cujos quais influenciam no resultado final do cubo. Neste trabalho está sendo utilizado três tipos de penalidades diferentes. A primeira penalidade tem por objetivo penalizar os quadrados de cada face que estão posicionados errados. A orientação da cor de cada face é dada pelo quadrado central. Essa primeira penalidade tem peso um e multiplica esse peso pela quantidade total de quadrados posicionados de forma errada no cubo.

A segunda penalidade tem por objetivo penalizar bordas que estão incorretamente posicionadas. Uma borda está corretamente posicionada quando existe um quadrado que liga a borda ao centro do cubo. De uma forma mais simples, o posicionamento de uma borda é considerado correto, quando esta possui em uma de suas respectivas faces o quadrado situado no meio da bor com a mesma cor que o quadrado situado no centro da face. A penalidade para posicionamento errado de bordas possui peso 4. Como um cubo possui 12 bordas, o máximo que essa penalidade pode atingir é 48.

A terceira penalidade tem por objetivo penalizar as quinas do cubo que estão incorretamente posicionadas. Uma quina está corretamente posicionada quando suas três são iguais as cores que estão posicionadas no centro de cada uma das faces que a mesma tem acesso. Essa penalidade possui peso 6, e como um cubo possui 8 quinas, o máximo que essa punição pode atingir é 48 também.

De acordo com essa Fitness, a penalidade máxima que um cubo pode possui é 144, uma vez que cada punição isolada atinge um máximo de 48. Assim, como são três tipos diferentes, 3x48 = 144.

**2.3. Cruzamento e Mutação**

Cruzamento e Mutação são dois tipos de operadores essenciais em um algoritmo genético, pois são eles os responsáveis pela evolução das soluções ao longo das gerações. Cada um desses operadores é descrito separadamente a seguir, iniciando pelo operador de cruzamento.

**2.3.1. Cruzamento**

Para aplicar o operador de cruzamento sempre é necessário a escolha de pais, os quais originam dois filhos, por meio de uma permutação de genes entre os dois ancestrais. A forma com que a permutação é realizada, varia de algoritmo para algoritmo. Neste trabalho, o cruzamento baseou-se na estratégia de ponto de corte. Essa estratégia consiste em escolher aleatoriamente uma posição do vetor em que o genótipo está situado, e esta é utilizada como o ponto limite para a permutação. O método de seleção utilizada para escolha dos pais candidatos a cruzarem é realizada por um torneio.

A seguir, é mostrado um exemplo de estratégia de cruzamento por meio de ponto de corte. Para um melhor entendimento, considere para a ilustração a seguir, um vetor de tamanho 10 e com um índice de ponto de corte, selecionado aleatoriamente, igual a 4.

ind1 { genótipo = [F,B,D,F1|U,L,B2,F2,R,U2]

ind2 { genótipo = [D,U1,F,U|B1,U,F2,L,R,U1]

Após a permutação dos pais, tem-se:

ind3 { genótipo = [F,B,D,F1|B1,U,F2,L,R,U1]

ind4 { genótipo = [D,U1,F,U|U,L,B2,F2,R,U2]

**2.3.2. Mutação**

A mutação é um tipo de operador utilizado para proporcionar diversidade em uma população. A mutação utilizada para diversificar os indivíduos foi a mutação uniforme. Para este tipo de mutação é verificado a probabilidade de cada gene pertencente a um cromossomo sofrer mutação. O gene apenas sofre mutação se ele for selecionado. Quando isso acontece, é selecionado um outro gene de forma aleatória e substituído pelo atual.

A seguir, é mostrado um exemplo de estratégia de mutação uniforme. Para um melhor entendimento, considere para a ilustração a seguir, um vetor de tamanho 10 e que os índices: 4, 6 e 9, foram selecionados para sofrer mutação.

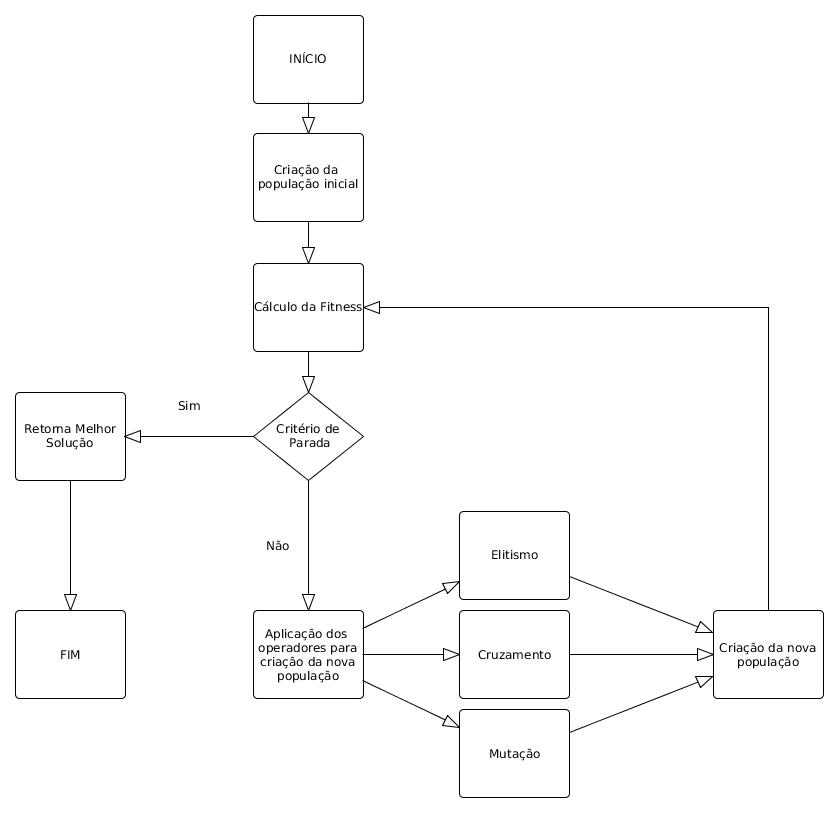
ind1 { genótipo = [F,B,D,F,**U**,L,**B2**,F2,R,**U2**]

Após a ocorrência da mutação, tem-se:

ind2 { genótipo = [F,B,D,F,**R**,L,**D1**,F2,R,**B**]

**2.4. Algoritmo Genético**

O algoritmo proposto é relativamente bem simples e seu funcionamento pode ser resumido nos seguintes passos ilustrados na **Figura 3**.



]

**Figura 3.** Fluxograma do passo a passo do algoritmo genético.

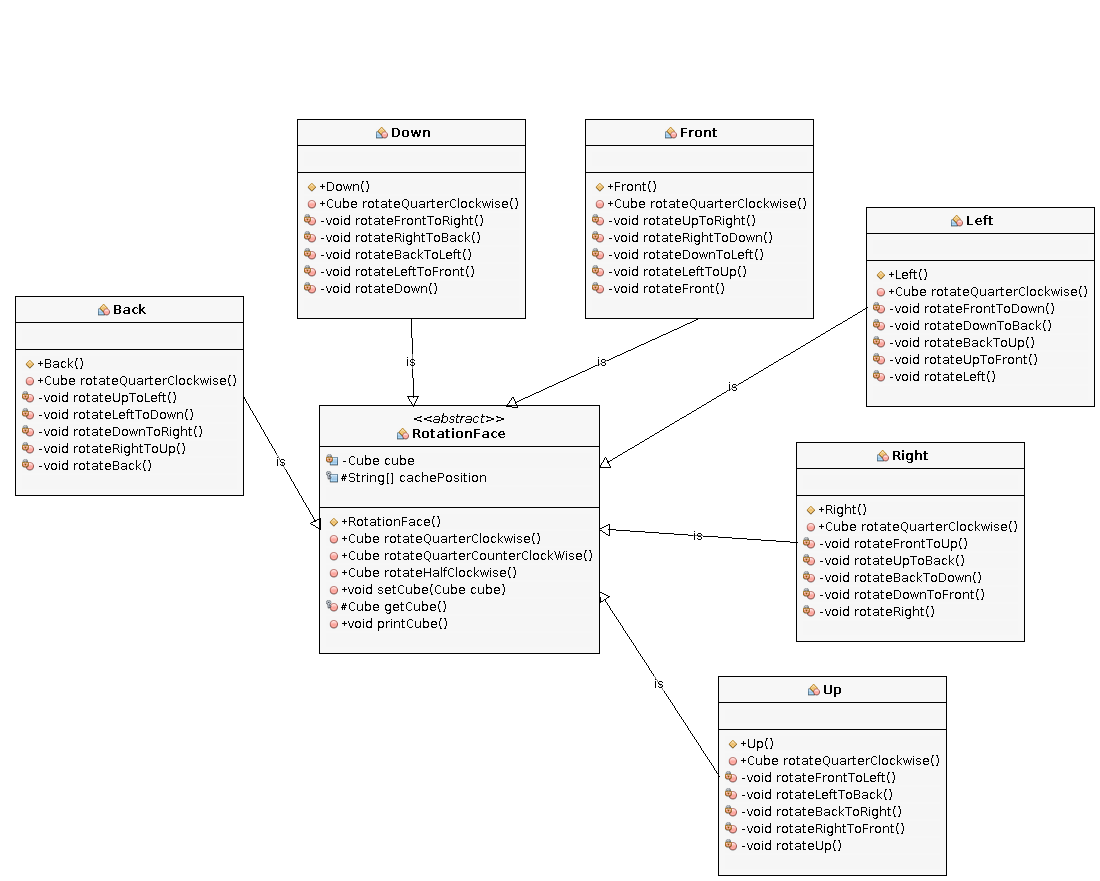
O algoritmo é bem tradicional. A solução implementada leva em consideração alguns parâmetros como: probabilidade de mutação, probabilidade de cruzamento, tamanho da população, dentre outros. A seção 4 possui uma maior detalhamento desses parâmetros, bem como instruções para execução do programa.

**3. Implementação**

A implementação do algoritmo genético foi realizada na linguagem Java. Como foi utilizada uma linguagem orientada a objetos, vários modelos foram representados em classes como: as rotações da face do cubo, os movimentos possíveis de serem realizados, o cromossomo, dentre outros. A implementação total resultou em 7 pacotes. Para uma melhor representação da implementação, é apresentado a seguir, os diagramas de classes desse algoritmo que estão organizados por pacotes.

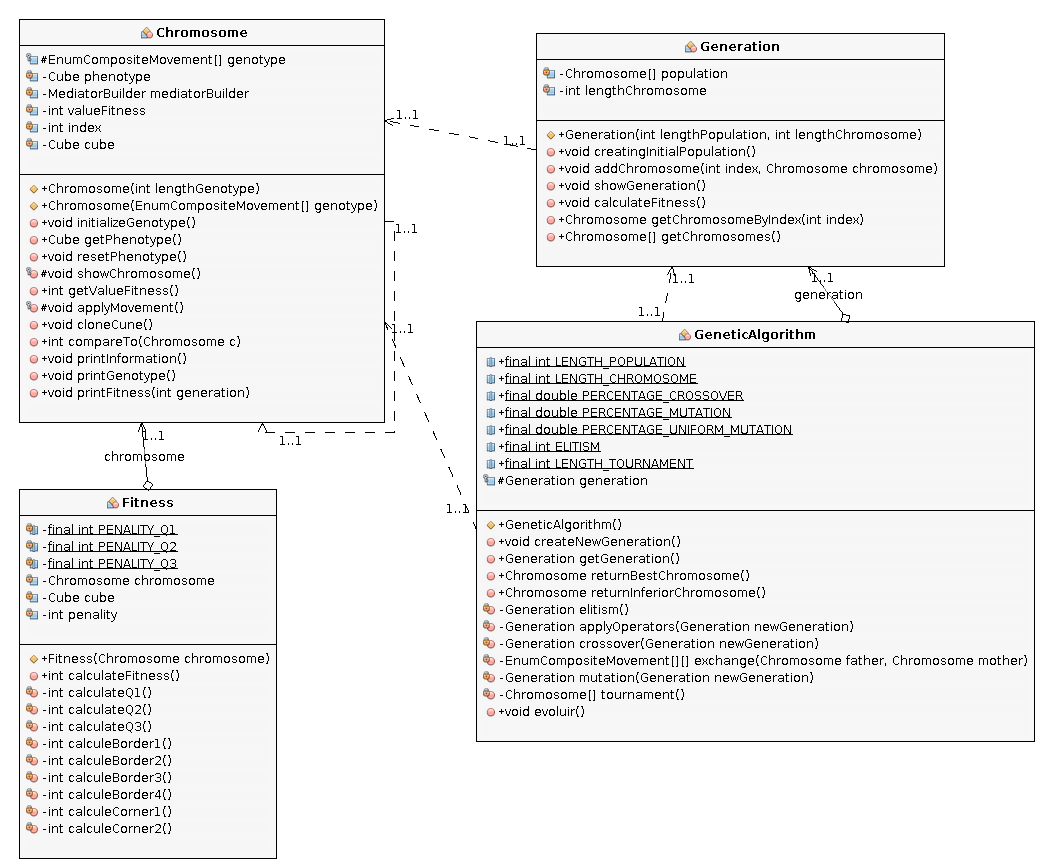
**3.1. Pacote rotation**

O pacote rotation contém as classes responsáveis por realizar as rotações de sentido 90º horário, 90º anti-horário e 180º horário em cada uma das seis faces existentes no cubo.

**Figura 4.** Diagrama de Classes referente ao pacote rotation.

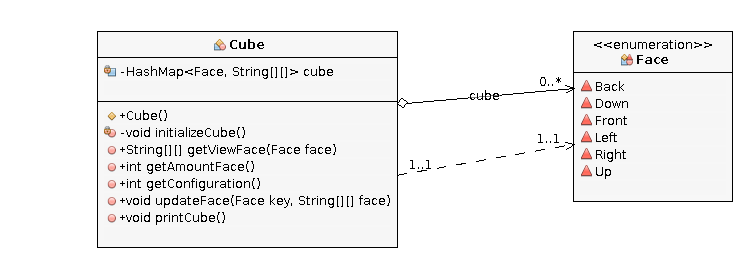
**3.2. Pacote structure**

O pacote structure contém as classes responsáveis pela estrutura do algoritmo genético. Para um melhor representação, foi criada uma classe para a representação da função Fitness, cromossomo, geração e algoritmo genético. As funcionalidades de cruzamento, mutação, elitismo e torneio, foram modeladas como métodos dentro da classe Genetic Algorithm.

**Figura 5.** Diagrama de Classes referente ao pacote structure.

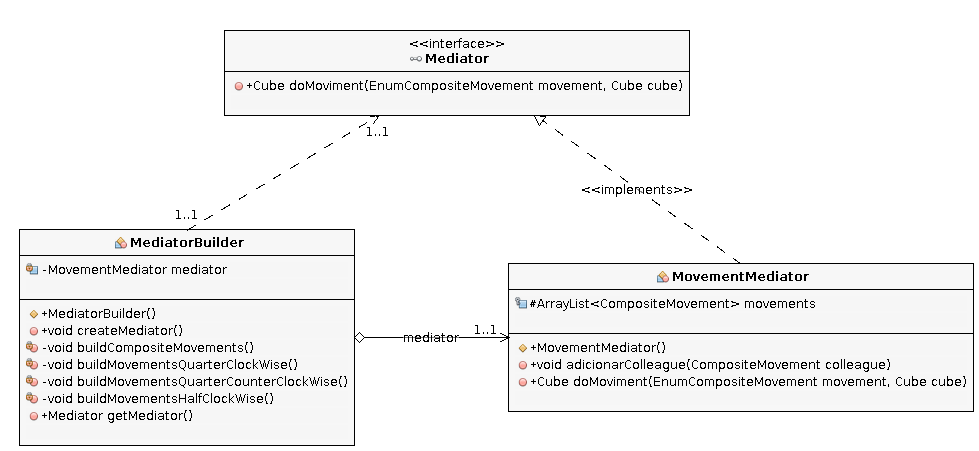
**3.3. Pacote structure.cube**

O pacote structure.cube contém as classes responsáveis pelo cubo. Neste pacote, o cubo é modelado em uma classe, e ainda existe um enum, referente as faces do cubo, para facilitar no gerenciamento do cubo.

**Figura 6.** Diagrama de Classes referente ao pacote structure.cube.

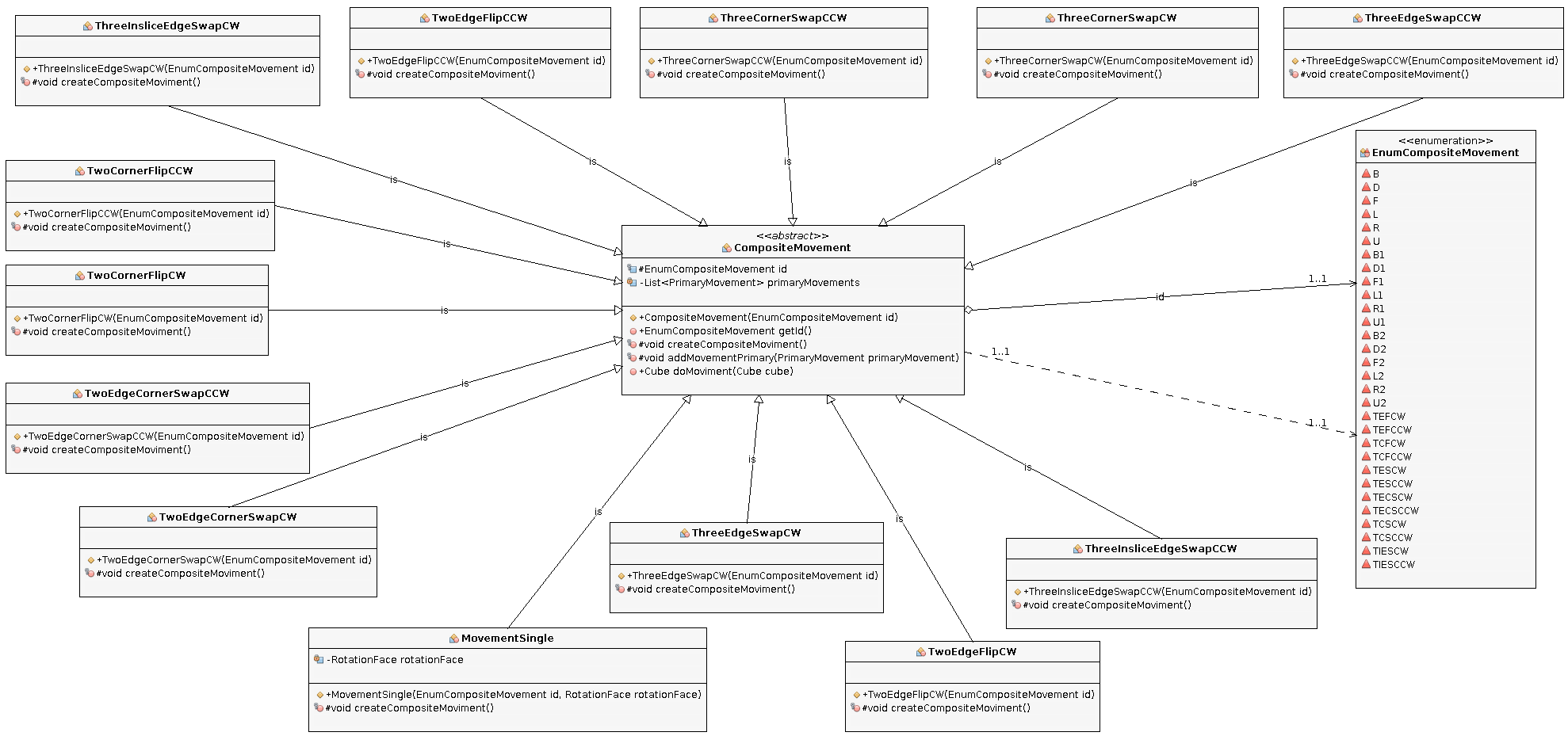
**3.4. Pacote structure.cube.movements**

O pacote structure.cube.movements contém as classes responsáveis por controlar os movimentos do cubo. Para realizar esse controle, foi utilizado o Padrão de Projeto mediator e builder. Esse pacote contém a classe referente ao padrão de projeto Builder (MediatorBuilder), e ainda possui duas classes referentes ao padrão de projeto mediator (Mediator e MovementMediator). A outra classe referente ao padrão de projeto mediator está localizada em outro pacote.

**Figura 7.** Diagrama de Classes referente ao pacote structure.cube.movements.

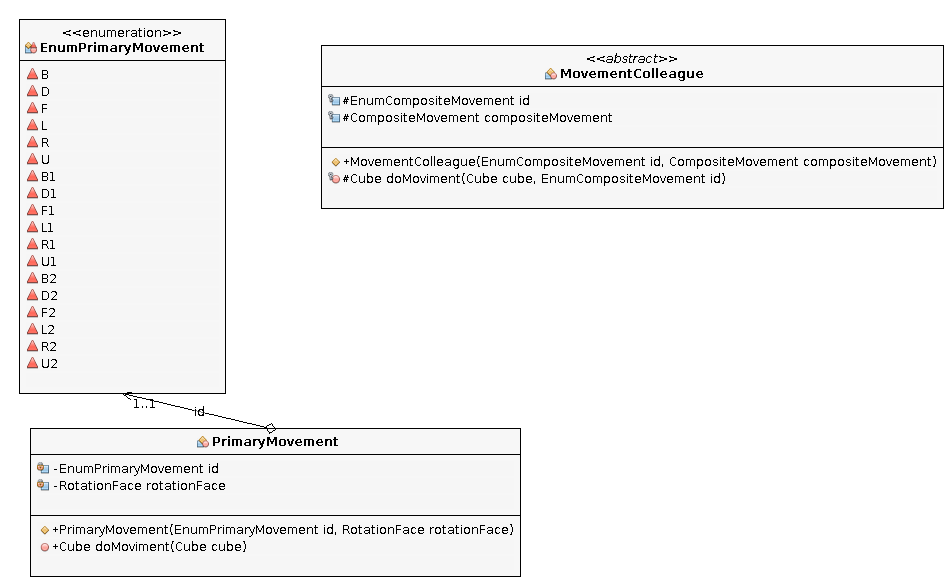
**3.5. Pacote** **structure.cube.movements.composite**

O pacote structure.cube.movements.composite contém as classes responsáveis por controlar os movimentos do cubo.

**Figura 8.** Diagrama de classes do pacote structure.cube.movements.composite

**3.6. Pacote structure.cube.movements.primary**

O pacote structure.cube.movements.primary contém as classes responsáveis por controlar os movimentos primários do cubo. A função delas é acionar as classes de rotação das faces de acordo com o comando acionado. Ainda, nesse pacote está presente a outra classe que compõe o padrão mediator (MovementColleague).

**Figura 9.** Diagrama de Classes do pacote structure.cube.movements.primary.

**4. Instruções para Execução do Algoritmo**

**Referências**

Herdy, M., Patone, G.: Evolution Strategy in Action, 10 ES-Demonstrations. Technical Report, International Conference on Evolutionary Computation (1994)