

# Documento do projeto Máquina de Estados

Mel Nakagawa Telesca

Gabriel Cunha Aita

Gabriel Pinheiro Franke

## 1. Visão geral do Cenário

Nosso trabalho se contextualiza em uma mineradora, há um minerador onde o mesmo pode ir minerar e gerar minérios, o minerador sempre vai descansar quando ele se cansa demais e por fim, se ele está com muito minérios ele vai entregar aos poucos para o vendedor. O vendedor inicialmente espera por minérios que serão entregues pelo minerador, assim que ele tiver algum minério, ele começa a vender aos poucos, até não sobrar nenhum minério em seu inventário, então ele volta a esperar por mais minérios.

## 2. Visão geral de cada agente

### 2.1. Minerador

O minerador terá as funções de minerar, entregar e descansar. Assim ele começará minerando, cada vez que minera ele gera 5 minérios, assim que ficar cansado (com 12 de cansaço) ele irá descansar até ficar totalmente descansado. Caso ele não se canse e tenha mais de 20 minérios no estoque ele irá começar a entregar os minérios para o vendedor (5 minérios por vez) até se cansar ou ficar sem minérios, o minerador pode entregar valores menores que 5 como 4, ou 2.

### 2.2. Vendedor

Enquanto o vendedor tiver minérios em seu estoque ele irá continuar vendendo os minérios (no máximo até 3 por vez) porém ele ainda pode vender 2 ou 1 minério até esgotar seu próprio estoque. Quando ele estiver sem minério ele volta a aguardar por mais minérios para vender.

## 3. Diagrama de estados por agente

Figura 1: Diagrama do Programa

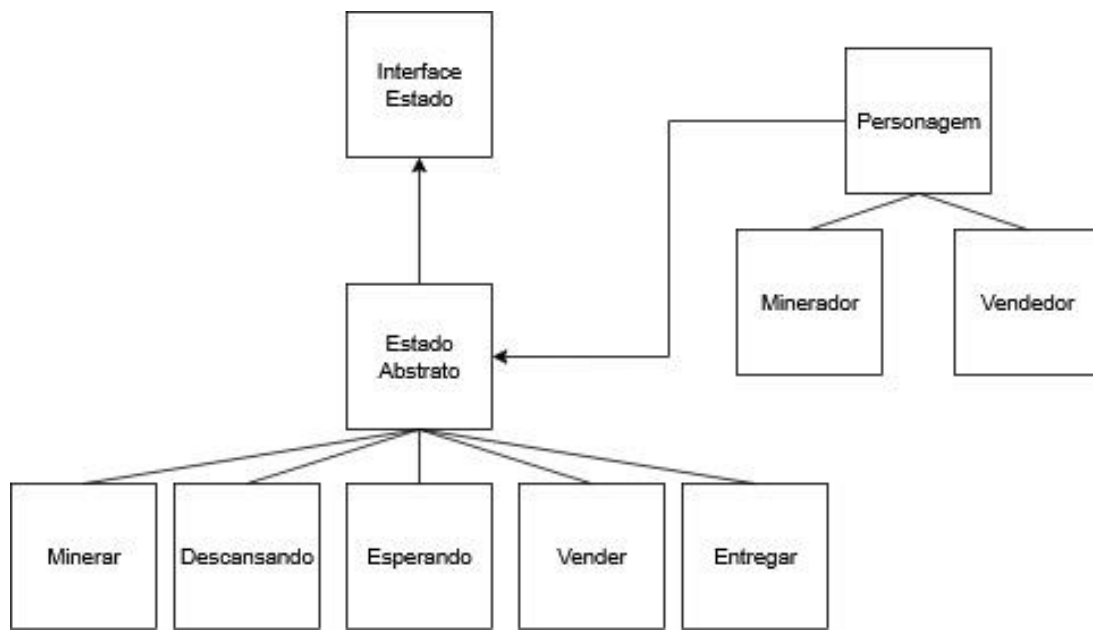


Figura 1: Criação Própria

#### 4. Tabela de regras por agente

Agente 1, Minerador:

Estado	Ação de entrada	Ação de saída	Execução	Transição
Minerar	Printa "Minerador foi minerar..."	Não há.	Soma 5 de minério e 2 de cansaço. Então printa os status do minerador.	Se cansaço for maior que 12, troca para Descansar. A segunda troca é se a quantidade de minério ser maior que 20, troca para Entregar.
Descansar	Printa "Minerador foi descansar..."	Printa "O minerador descansou, voltando ao trabalho!"	Reduz o cansaço em 4 pontos. E então printa os status do minerador.	Se cansaço ser igual a 0 e quantidade de minérios ser maior que 20 troca para Entregar. A segunda condição é se caso o cansaço ser igual a 0 e a quantidade de minério ser menor que 20, troca para Minerar.
Entregar	Printa "Minerador saiu para entregas..."	Não há.	Reduz em até 5 o minério do minerador e aumenta na mesma quantidade o minério do vendedor. Aumenta em 1 o cansaço. Então printa os status.	Se cansaço ser maior que 12, então troca para Descansar. A segunda condição é se a quantidade de minério ser igual a 0, então troca para Minerar.

Agente 2, Vendedor:

Estado	Ação de entrada	Ação de saída	Execução	Transição
--------	-----------------	---------------	----------	-----------

VenderMinerio	Printa "Vendedor começou a vender os minérios..."	Printa "Entregas realizadas!"	Vende até 3 minérios por vez. Então printa os status do vendedor.	Se a quantidade de minério do vendedor for igual a 0, então troca para EsperarMinerio.
EsperarMinerio	Printa "O vendedor agora está aguardando algum minério ser entregue..."	Não há.	Printa "O vendedor agora está aguardando algum minério ser entregue..."	Se a quantidade de minério do vendedor for maior que 0, então troca para VenderMinerio.

## 5. Descrição das variáveis

Minerador – Terá as variáveis:

int Cansaço, valor inicial 0. Acima de 12 o minerador irá descansar.

int minérios, valor inicial 0. Acima de 20 o minerador irá começar a entregar os minérios para o vendedor.

Vendedor vendedor, a referência do vendedor que o minerador irá entregar.

EstadoAbstrato estado, a referência do estado atual do minerador, inicial é Minerando.

Vendedor – Terá as variáveis:

int minerioEstocado, inicialmente 0. É a quantidade de minério que o vendedor tem, acima de 0 ele começa a fazer entregas.

int minerioVendido, a quantidade de minérios que o vendedor realizou na passagem, o máximo é 3, mas pode ser 2 ou 1. Até que todos os minérios sejam entregues, então ele sempre será 0 (pois logicamente nenhum minério foi vendido).

EstadoAbstrato estado, é a referência do estado atual do vendedor. Inicialmente é EsperarMinerios.

## 6. Estrutura do código

Arquivos- Os arquivos do projeto possuem os mesmos nomes que as classes e estão todos no src.

Classes-

GerenciadorAgentes: Ele cria uma lista para guardar os agentes e possui uma função que cria os agentes e então em um while chama a função executar dos agentes.

Personagem: A interface que define algumas regras gerais para qualquer agente.

Estados: A interface que define as regras gerais para a classe abstrata dos estados.

EstadoAbstrato: A classe abstrata que define métodos importantes como getPersonagem, o construtor e entrar, executar e sair.

Minerador: a classe que define o agente que irá minerar. Nela há as variáveis “minerios” e “cansaco” bem como set e get delas. Além disso temos o construtor e referências para o estado atual, a referência do vendedor. Temos também a função “fazer” que executa a função dos estados.

Vendedor: a classe que define o agente que irá vender os minérios, ela possui as variáveis de “minerioEstocado” e “minerioVendido”. O vendedor irá vender o minerioEstocado e alterar o minerioVendido. E por fim possui a variável “estado” que guarda o estado atual do vendedor. Também tem gets e sets, além da função “fazer” que executa a função dos estados.

Minerar: a classe concreta do EstadoAbstrato que faz o Minerador Minerar, aumentando o minério e o cansaço.

Descansar: classe concreta do EstadoAbstrato que faz o Minerador Descansar e diminuir o cansaço.

Entregar: a classe concreta do EstadoAbstrato que faz o Minerador passar parte dos minérios dele para o Vendedor.

EsperarMinerios: classe concreta do EstadoAbstrato que faz o Vendedor esperar até que receba ao menos um minério.

VenderMinerios: classe concreta do EstadoAbstrato que reduz o valor do minério do Vendedor e define quantos minérios foram vendidos.

## 7. Resultados esperados

Para o output inicial do minerador temos como esperado algo semelhante a isso:

```
+---- Minerador ----+
```

Minérios: 5

Cansaço: 2

Como ele começa minerando, ele na primeira passagem gera 5 minérios e cansa no valor de 2 pontos.

Logo abaixo temos o primeiro print do vendedor:

+---- Vendedor ----+

O vendedor está esperando minérios...

Como o vendedor começa esperando e sem minérios, ele apenas diz que está aguardando algum minério para ele começar a trabalhar.

O código segue até o momento que a quantidade de minérios passa de 20 e assim ele vai fazer entregas, o log retornará:

+---- Minerador ----+

Minérios: 25

Cansaço: 10

Minerador saiu para entregas...

Quando o cansaço ser maior que 12, então será impresso algo semelhante a:

+---- Minerador ----+

Minérios: 10

Cansaço: 13

O minerador foi descansar...

Já quando o vendedor receber sua primeira entrega, ele trocará seu modo para começar as vendas:

+---- Vendedor ----+

O vendedor está esperando minérios...

Vendedor começou a vender os minérios...

Assim que ele fizer as vendas de fato, o log imprimirá os status.  
Como ele recebe 5 minérios por vez e ele vende no máximo 3, o console deverá retornar como a primeira venda:

+---- Vendedor ----+

Minérios vendidos: 3

Minérios no estoque: 7

Quando ele vende todos os minérios ele volta a aguardar como no exemplo:

+---- Vendedor ----+

Minérios vendidos: 3

Minérios no estoque: 0

Entregas realizadas!

O vendedor agora está aguardando algum minério ser entregue...

## 8. Limitações (Opcional)

O grupo conseguiu implementar toda a ideia com sucesso. O maior desafio foi entender sobre a Tag que representa a função genérica e quando ela passa a valer para o tipo da classe de referência.

## 9. Referências (Opcional)

O trabalho foi feito de forma original, mas com algumas inspirações da atividade de Juca vista em sala. Por exemplo usar uma variável que representa o cansaço e um estado que serve para descansar.