

# Máquina de Estados – Marco e Pollo

Feito por João Vitor Faria Crema e Pedro Baldissera

## 1. Visão Geral

Este projeto implementa uma Máquina de Estados em Java com dois agentes: Marco e Pollo. Cada agente possui comportamentos distintos baseados em seus estados internos, e ambos compartilham um ponto de interação em comum, simulando comunicação e tomada de decisão dentro do loop principal.

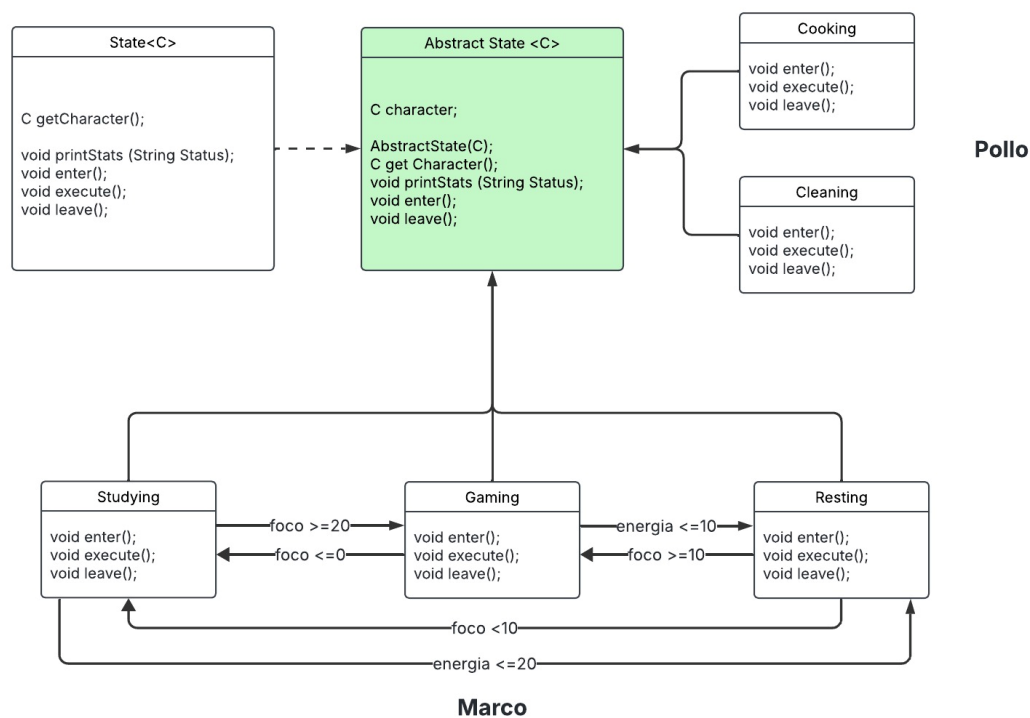
## 2. Agentes

**Marco:** Representa um agente mais ativo e comunicativo. Ele pode alternar entre estados como Trabalhando, Dormindo e Comendo.

**Pollo:** Um agente mais observador e cauteloso, com os mesmos estados básicos, mas respostas diferentes em relação ao ambiente e às ações de Marco

## 3. Diagrama de estados por agente

Cada agente possui uma máquina de estados independente que representa seu comportamento em diferentes situações. Esses estados definem o que o agente faz em determinado momento e quais condições provocam a transição entre eles.



## 4. Tabela de Regras por Agente

### Agente Marco

O agente Marco possui três estados principais:

- **Estudando**
  - Entrada: inicia o estudo, aumentando o foco.
  - Execução: mantém o foco elevado, mas reduz a energia gradualmente.
  - Saída: quando a energia cai abaixo do limite definido, o estado muda para *Descansando*.
  - Transição:
    - $Energia < 30 \rightarrow$  vai para descansando
    - $Foco > 70 \rightarrow$  pode alternar para jogando
- **Descansando**
  - Entrada: o agente relaxa para recuperar energia.
  - Execução: aumenta a energia aos poucos.
  - Saída: ao atingir energia suficiente, decide o próximo estado com base no foco.
  - Transição:
    - $Energia > 80$  e  $Foco < 50 \rightarrow$  Jogando
    - $Energia > 80$  e  $Foco \geq 50 \rightarrow$  Estudando
- **Jogando**
  - Entrada: inicia o lazer, reduzindo foco, mas aumentando energia.
  - Execução: relaxa e regenera energia.
  - Saída: quando o foco aumenta novamente, volta a estudar.
  - Transição:
    - $Foco \geq 60 \rightarrow$  Estudando

## Agente Pollo

O agente Pollo apresenta dois estados principais:

### Trabalhando

- Entrada: começa o trabalho, reduzindo energia e aumentando fome.
- Execução: continua produtivo até atingir os limites de energia ou fome.
- Saída: ao ficar com energia baixa ou fome alta, transita para *Comendo*.
- Transição:
  - $Energia < 40$  ou  $Fome > 60 \rightarrow Comendo$

- **Comendo**

- Entrada: inicia a refeição, reduzindo a fome e recuperando energia.
- Execução: come até atingir o limite ideal.
- Saída: quando está saciado e com energia suficiente, volta ao trabalho.
- Transição:
  - $Fome \leq 20$  e  $Energia \geq 60 \rightarrow Trabalhando$

## 5. Variáveis e Limiares por Agente

### Agente Marco

- **Energia:** varia de 0 a 100.
  - *Limite inferior (30):* transita para descanso.
  - *Limite superior (80):* pode voltar a estudar ou jogar.
- **Foco:** varia de 0 a 100.
  - *Limite superior (70):* alterna para lazer.
  - *Limite inferior (50):* volta ao estudo.

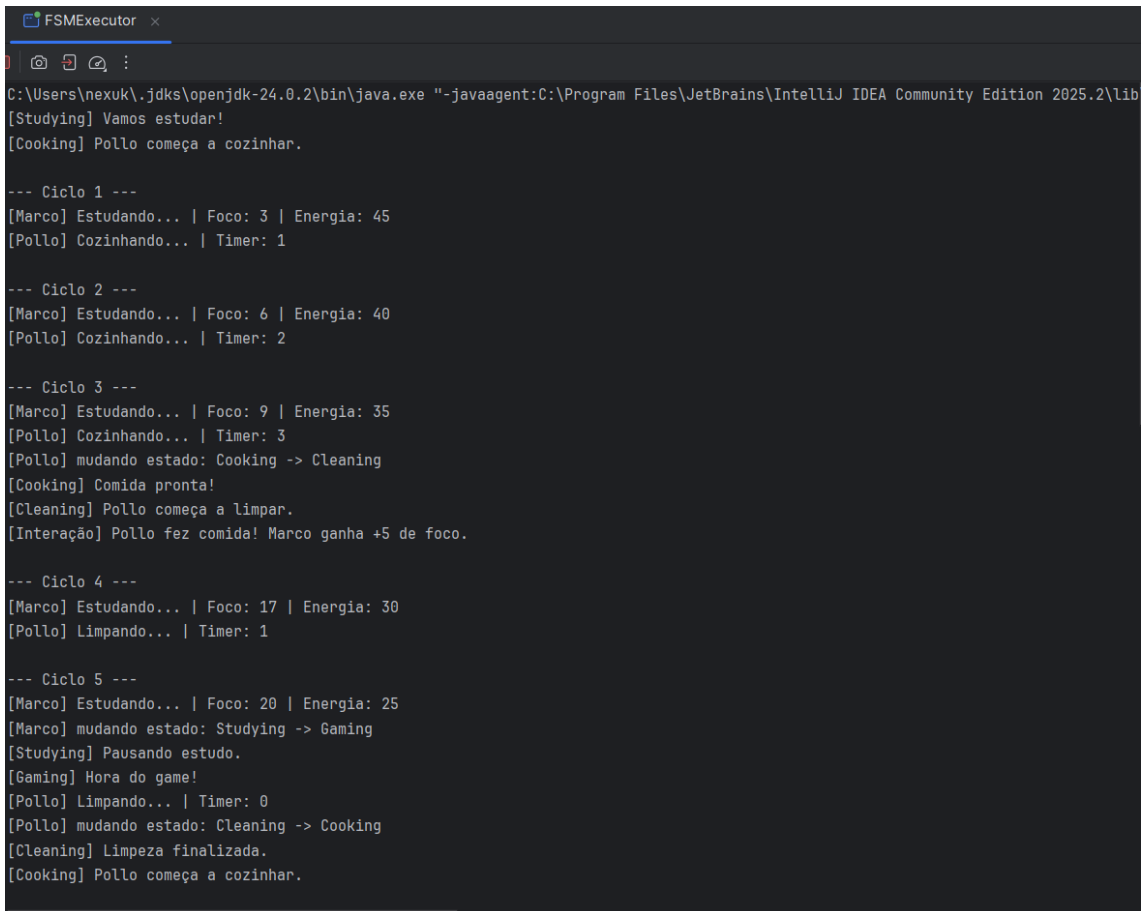
## Agente Pollo

- **Energia:** varia de 0 a 100.
  - *Limite inferior (40):* pausa o trabalho.
  - *Limite superior (60):* retorna ao trabalho.
- **Fome:** varia de 0 a 100.
  - *Limite superior (60):* começa a comer.
  - *Limite inferior (20):* volta ao trabalho.

## 6. Estrutura do Código e Funcionamento

- **Classe State (interface):**  
Define os métodos essenciais de um estado: `enter()`, `execute()` e `exit()`.
- **Classe AbstractState:**  
Implementa parte da interface State e serve como base para os estados específicos (ex.: Trabalhando, Descansando, Estudando).
- **Classes de Estado (Working, Eating, Sleeping, etc.):**  
Representam ações específicas do agente, contendo a lógica de transição entre os estados.
- **Classe Marco e Pollo:**  
Cada uma representa um agente independente, com variáveis próprias (como energia, foco ou fome) e seus respectivos estados.
- **Classe Main:**  
É o ponto de entrada do programa. Cria os agentes e executa um loop que atualiza o comportamento deles conforme suas condições mudam.

## 7. Logs e Resultados



```
FSMExecutor x
C:\Users\nexuk\.jdk\openjdk-24.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2025.2\lib
[Studying] Vamos estudar!
[Cooking] Pollo começa a cozinhar.

--- Ciclo 1 ---
[Marco] Estudando... | Foco: 3 | Energia: 45
[Pollo] Cozinhando... | Timer: 1

--- Ciclo 2 ---
[Marco] Estudando... | Foco: 6 | Energia: 40
[Pollo] Cozinhando... | Timer: 2

--- Ciclo 3 ---
[Marco] Estudando... | Foco: 9 | Energia: 35
[Pollo] Cozinhando... | Timer: 3
[Pollo] mudando estado: Cooking -> Cleaning
[Cooking] Comida pronta!
[Cleaning] Pollo começa a limpar.
[Interação] Pollo fez comida! Marco ganha +5 de foco.

--- Ciclo 4 ---
[Marco] Estudando... | Foco: 17 | Energia: 30
[Pollo] Limpando... | Timer: 1

--- Ciclo 5 ---
[Marco] Estudando... | Foco: 20 | Energia: 25
[Marco] mudando estado: Studying -> Gaming
[Studying] Pausando estudo.
[Gaming] Hora do game!
[Pollo] Limpando... | Timer: 0
[Pollo] mudando estado: Cleaning -> Cooking
[Cleaning] Limpeza finalizada.
[Cooking] Pollo começa a cozinhar.
```

## 8. Limitações

## 9. Referencias

Usamos de exemplo o Juca apresentado em sala de aula

<https://www.w3schools.com>

### GitHub:

<https://github.com/Nexukk/Maquina-de-Estados/tree/main>

### ReadME:

Disponível no Github.