Trabalho de Programação Orientada a Objetos

**Nome:** Geovane Neves Armelini

**Nome:** Giann Augusto Antunes de Oliveira

# Organização do Projeto com Scrum

O desenvolvimento do projeto foi guiado pelo framework Scrum, dividido em duas Sprints semanais. A primeira Sprint foi voltada à construção da base do sistema utilizando princípios fundamentais de orientação a objetos, como herança, encapsulamento e uso de interfaces. A segunda Sprint concentrou-se em refatoração e aplicação de polimorfismo, além da realização de testes.

## 1ª Sprint: Estruturação Inicial

Nesta etapa, foram criadas as estruturas fundamentais do projeto:

- Superclasse `Aves`: define atributos e comportamentos gerais compartilhados por todas as aves.

- Interfaces `AveVoa` e `AveNaoVoa`: abstraem o comportamento de voo, permitindo flexibilidade na implementação.

- Classes `Pato` e `Pinguim`: cada uma implementa os comportamentos de forma distinta, utilizando herança da superclasse `Aves` e implementando as interfaces adequadas.

- Encapsulamento aplicado: todos os atributos foram protegidos com `private`, e o acesso foi controlado por métodos `get` e `set`.

## 2ª Sprint: Polimorfismo e Refatoração

Com a estrutura pronta, iniciou-se a fase de melhoria e testes:

- O método `bicar()` foi refatorado em `Pinguim`, aplicando polimorfismo para comportamentos distintos entre as subclasses.

- A substituição de acesso direto aos atributos foi feita por métodos como `setBico()`, promovendo boas práticas de encapsulamento.

- Os testes verificaram a diferenciação de comportamentos entre pato e pinguim, garantindo que o polimorfismo estivesse funcionando corretamente.

## Casos de Uso

Três casos de uso principais foram mapeados:

1. Controle de Bicada: Permitir que o usuário ative ou desative a ação de bicar em cada ave.

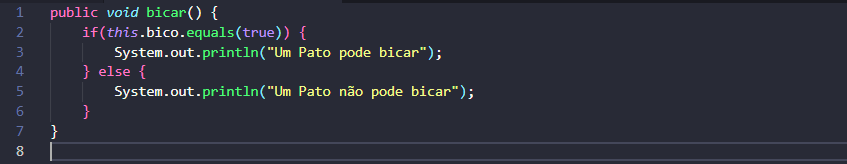
2. Comparação entre espécies: Destacar as diferenças de comportamento entre o pato (que voa) e o pinguim (que nada).

3. Herança e Polimorfismo: Mostrar como as subclasses especializam comportamentos herdados da superclasse `Aves`.

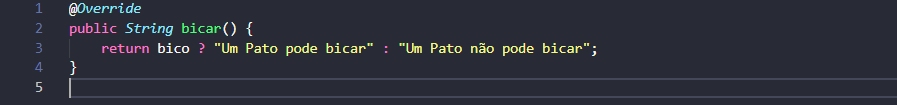
## Refatoração Aplicada

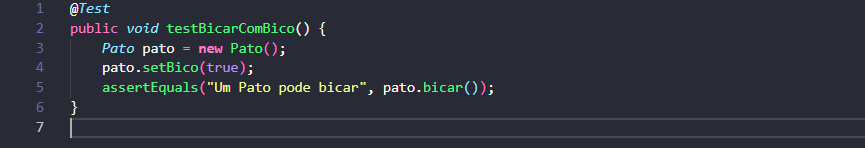
Durante o processo, a classe `Pinguim` foi revista para eliminar redundâncias e aplicar melhor clareza de código. O método `bicar()` foi personalizado para refletir as características dessa ave, respeitando o princípio de substituição de Liskov e tornando o código mais coeso.

Antes:



Depois:

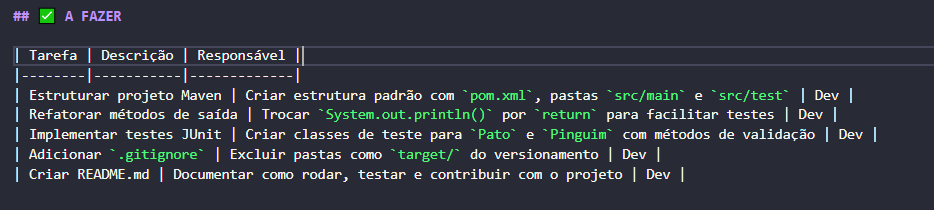




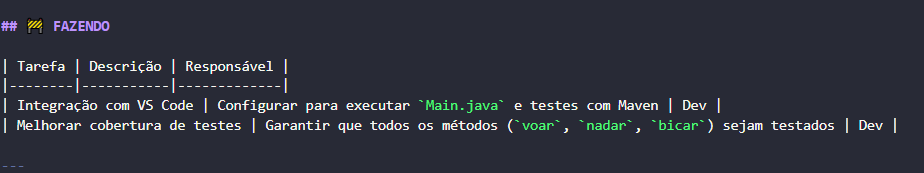
## Gerenciamento com Kanban

Um quadro no Trello foi utilizado para organizar as tarefas com a metodologia Kanban. As colunas foram divididas em:

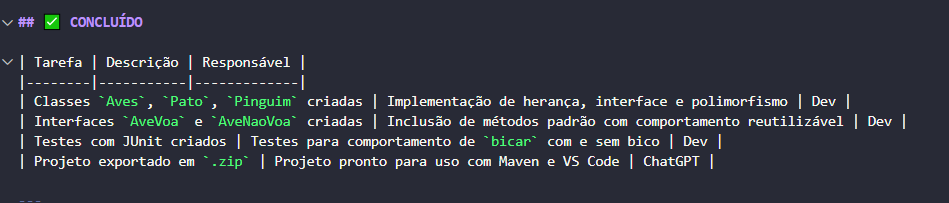
* A Fazer:



* Fazendo:



* Concluido:



Esse método facilitou a visualização do progresso e ajudou no cumprimento das entregas das Sprints.