Domain Driven Design - Java

Prof. Gilberto Alexandre das Neves
profgilberto.neves@fiap.com.br

Orientação à Objetos

História

Nos anos 60 nasceu a programação estruturada. Esse é o método estimulado por linguagens como C e Pascal.

Usando-se linguagens estruturadas, foi possível, pela primeira vez, escrever programas moderadamente complexos de maneira razoavelmente fácil.

Entretanto, com programação estruturada, quando um projeto atinge um certo tamanho, torna-se extremamente difícil e muito custoso efetuar sua manutenção e fazer qualquer modificação.

A primeira linguagem a incorporar facilidades para definir classes de objetos genéricos na forma de uma hierarquia de classes e subclasses foi a linguagem Simula, que foi idealizada em 1966, na Noruega.

Orientação à Objetos

A Programação Orientada à Objetos aproveitou as melhores ideias da programação estruturada e combinou-as com novos conceitos, permitindo que um problema seja mais facilmente decomposto em subgrupos relacionados. Então, usando-se a linguagem, pode-se traduzir esses subgrupos em objetos.

A Programação Orientada à Objetos utiliza os conceitos que aprendemos no jardim de infância: objetos e atributos, todos e partes, classes e membros.

No entanto, cabe ressaltar que o conceito de Orientação à Objetos depende mais da mentalidade do programador do que da linguagem de programação que está sendo utilizada. Tomemos como exemplo a frase:

"**O navio atraca no porto e descarrega sua carga**."

FIAP

"**O navio atraca no porto e descarrega sua carga**."

Se analisássemos esta frase estruturadamente, pensaríamos logo em como o navio atraca no porto e como ele faz para descarregar sua carga, ou seja, pensaríamos na ação que está sendo feita (que na frase é representada pelos verbos) para transformá-la em procedimento.

Em orientação objeto, o enfoque com que se encara a frase é diferente: primeiro pensaríamos no objeto navio, no objeto porto e no objeto carga, pensando como eles seriam e procurando definir seu comportamento. Após isto é que pensaremos em como o navio se relaciona com o porto e com a carga, e como o porto se relaciona com carga.

Conceitos da Orientação à Objetos

O mundo real é algo extremamente complexo. Quanto mais de perto o observamos, mais claramente percebemos sua complexidade.

A orientação a objetos tenta gerenciar a complexidade inerente dos problemas do mundo real, ***abstraindo*** conhecimento relevante e ***encapsulando-o*** dentro de objetos.

Abstração

Uma das principais formas do ser humano lidar com a complexidade é através do uso de abstrações.

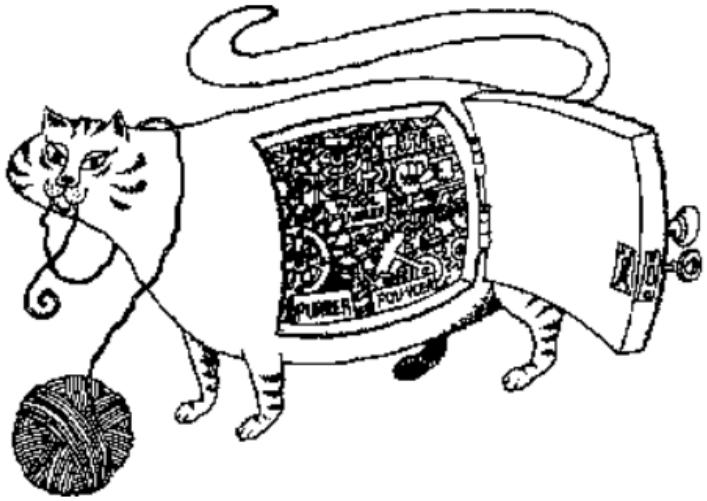
As pessoas tipicamente tentam compreender o mundo, construindo modelos mentais de partes dele. Tais modelos são uma visão simplificada de algo, onde apenas elementos relevantes são considerados.

Exemplo: Mapa de um território.

Encapsulamento

No mundo real, um objeto pode interagir com outro sem conhecer seu funcionamento interno. Uma pessoa, por exemplo, geralmente utiliza uma televisão sem saber efetivamente qual a sua estrutura interna ou como seus mecanismos internos são ativados. Para utilizá-la, basta saber realizar algumas operações básicas, tais como ligar/desligar a TV, mudar de um canal para outro, regular volume, cor, etc.

O encapsulamento consiste na separação dos aspectos externos de um objeto, acessíveis por outros objetos, de seus detalhes internos de implementação, que ficam ocultos dos demais objetos.



Modularidade

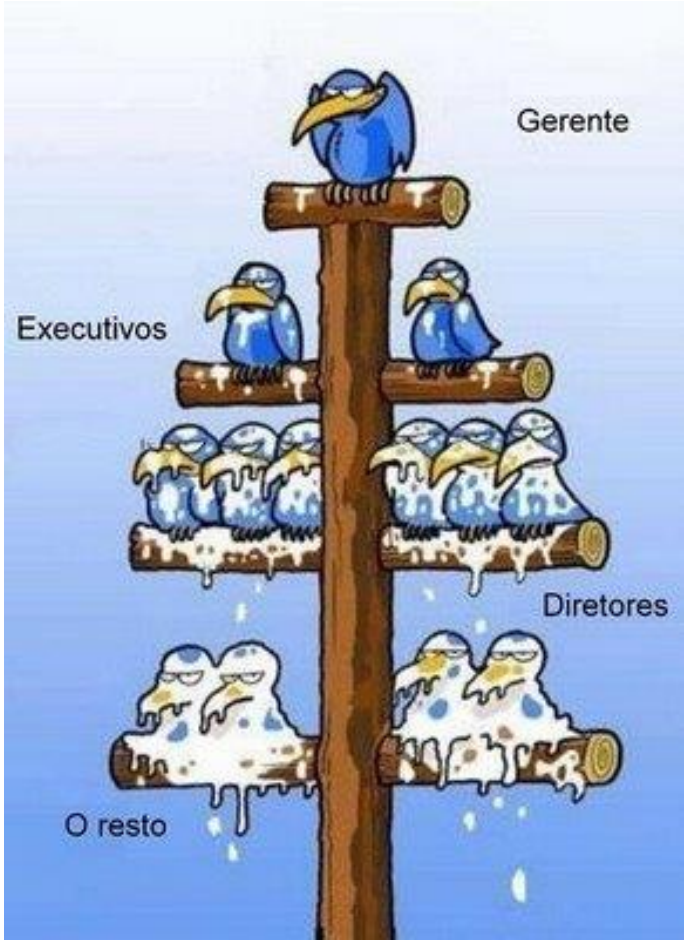
Muitos métodos de construção de software buscam obter sistemas modulares, isto é, construídos a partir de elementos que sejam autônomos, conectados por uma estrutura simples e coerente.

Modularidade é crucial para se obter re-usabilidade e extensibilidade.

Hierarquia

Um conjunto de abstrações frequentemente forma uma hierarquia e, pela identificação dessas hierarquias, é possível simplificar significativamente o entendimento sobre um problema.

Em suma, hierarquia é uma forma de arrumar as abstrações.



Classes e Objetos

Uma classe é um **molde** ou **modelo** que define as características (**atributos**) e os comportamentos (**métodos**) de um objeto.

Para entender melhor o conceito de classe, vamos analisar suas ***instâncias***, conhecidas como **objeto**. Um objeto é um termo que usamos para representar uma entidade do mundo real (fazemos isso através do exercício da abstração).

Vamos usar como exemplo o cachorro *Muttley.*

Classes e Objetos

Podemos representá-lo em termos de **atributos**:

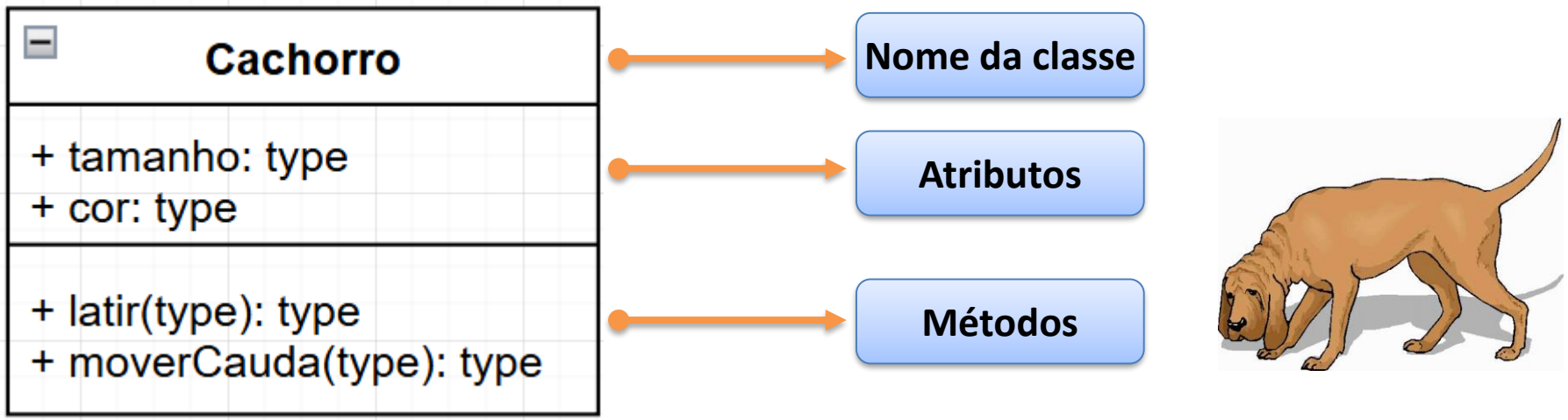
- Seu ***tamanho*** é pequeno
- Sua ***cor*** predominante é castanha

Podemos representá-lo também através de seu “comportamento” (**métodos**):

- Ele é capaz de ***latir***
- Ele é capaz de ***balançar a cauda***

Identificação de Classes

Logo, *Muttley* é um objeto da classe **Cachorro** da qual ele faz parte.



Atenção para a **convenção de nomenclatura** para os **atributos** e **métodos**, devem começar com letra minúscula e a primeira letra de cada nova palavra maiúscula (não deve ter espaços).

UML

A **UML** (Unified Modeling Language) é uma linguagem visual padronizada para modelar sistemas de software. Ela ajuda a representar a estrutura e o comportamento de um sistema de forma clara e compreensível.

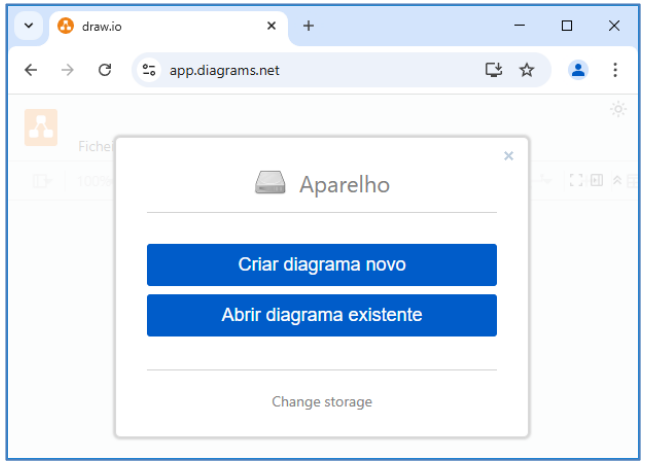
- **Objetivo**: Facilitar a comunicação entre desenvolvedores, arquitetos de software e *stakeholders*.
- **Principais Diagramas**: UML possui vários diagramas, como diagrama de classes, casos de uso, sequência, atividades, entre outros.

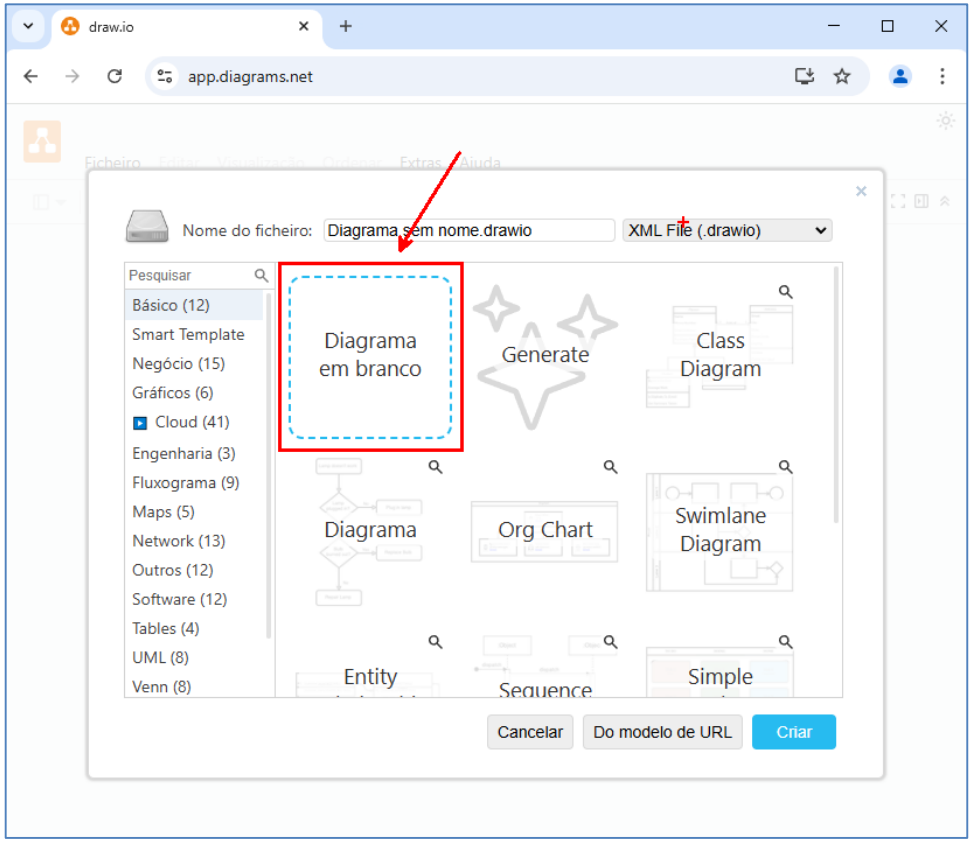
O **diagrama de classes** é um dos mais importantes da UML. Ele representa a estrutura estática do sistema, mostrando:

- ✓ Classes e seus atributos
- ✓ Métodos (comportamentos das classes)
- ✓ Relacionamentos entre as classes (associação, herança, agregação, composição)

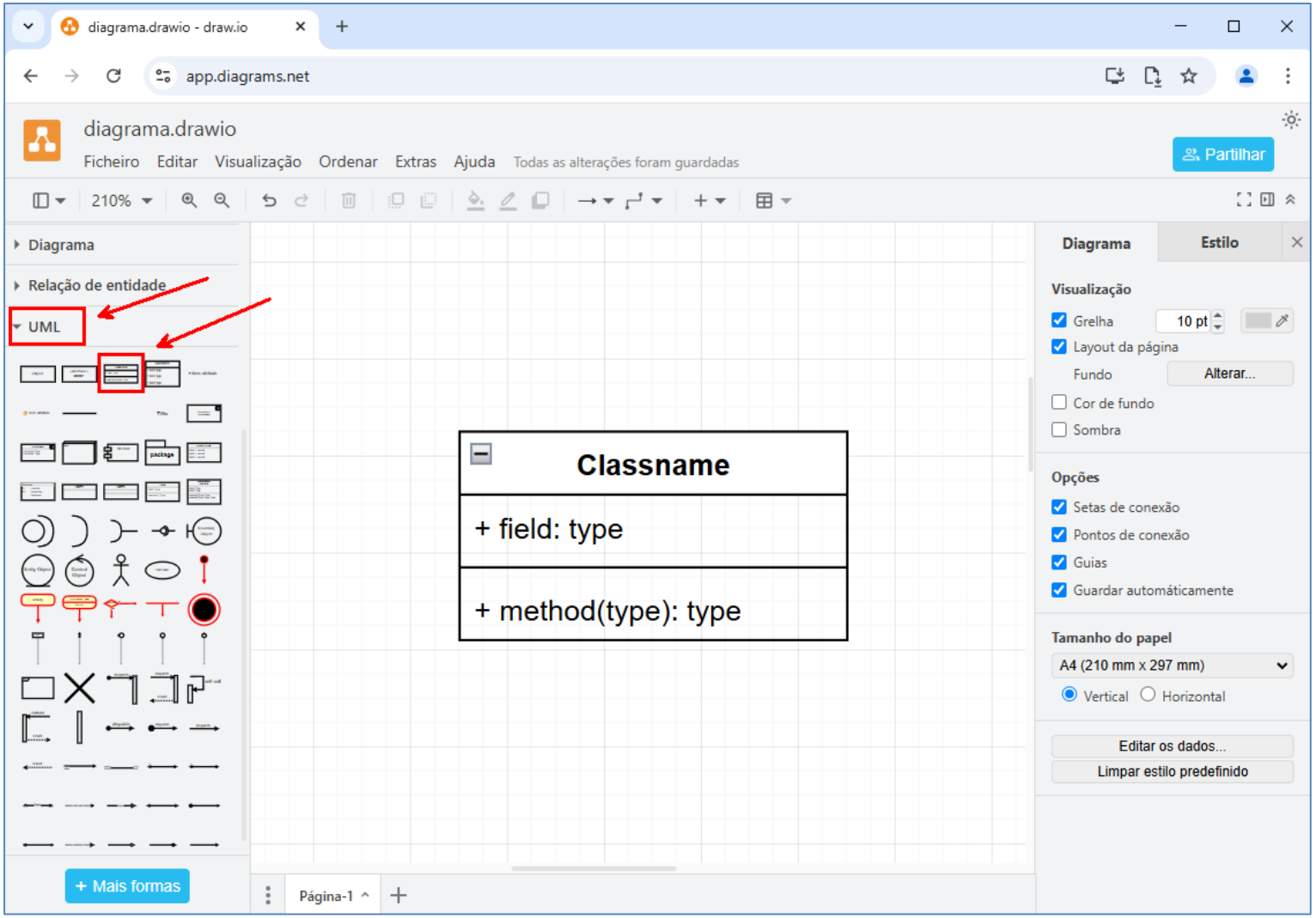
app.diagrams.net (draw.io)

Para desenvolver **diagrama de classes** com a linguagem **UML** recomendo o uso da ferramenta **app.diagrams.net** (antigo **draw.io**). Ferramenta de desenho de diagramas **online** que não necessita instalação.

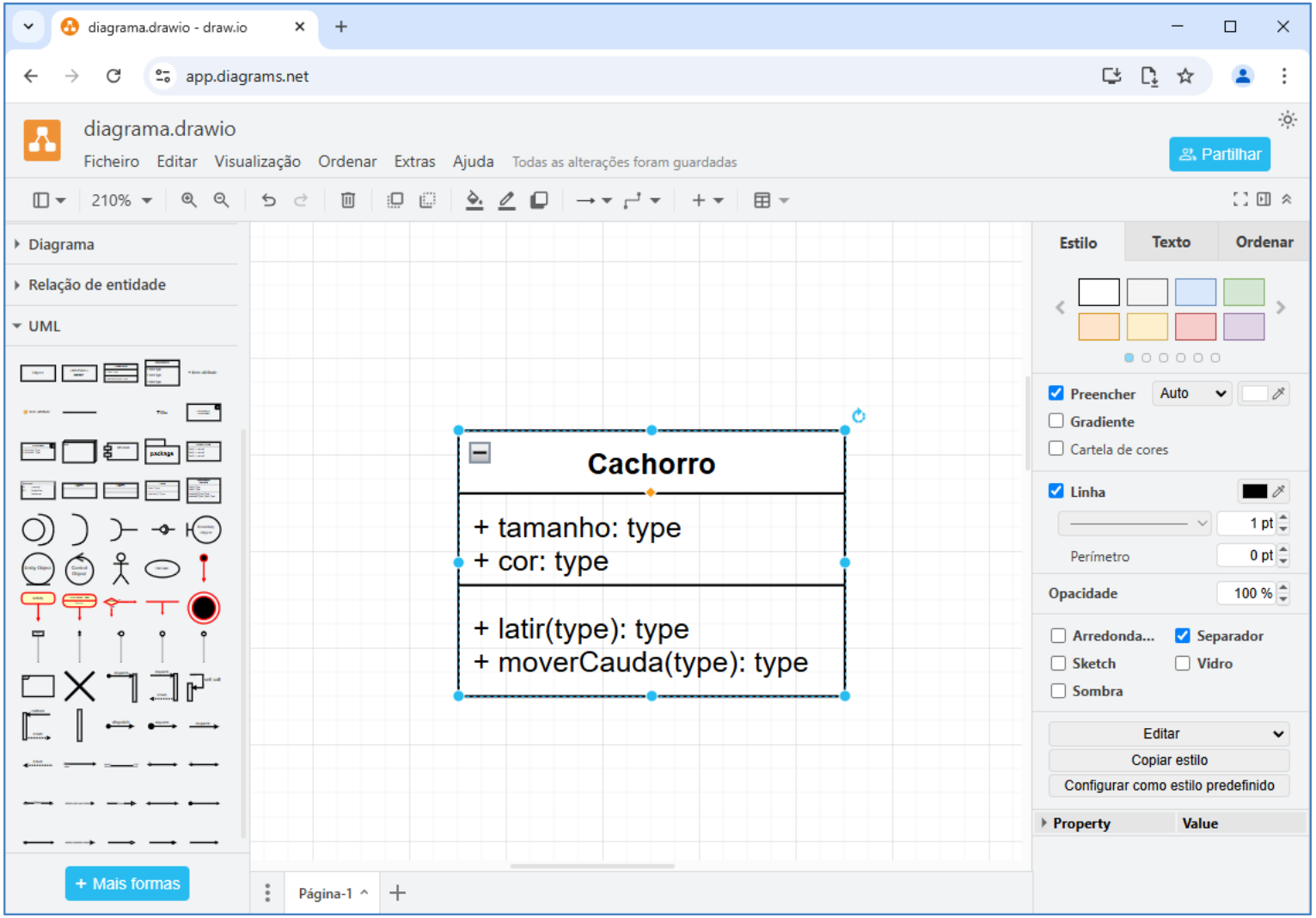




app.diagrams.net (draw.io)



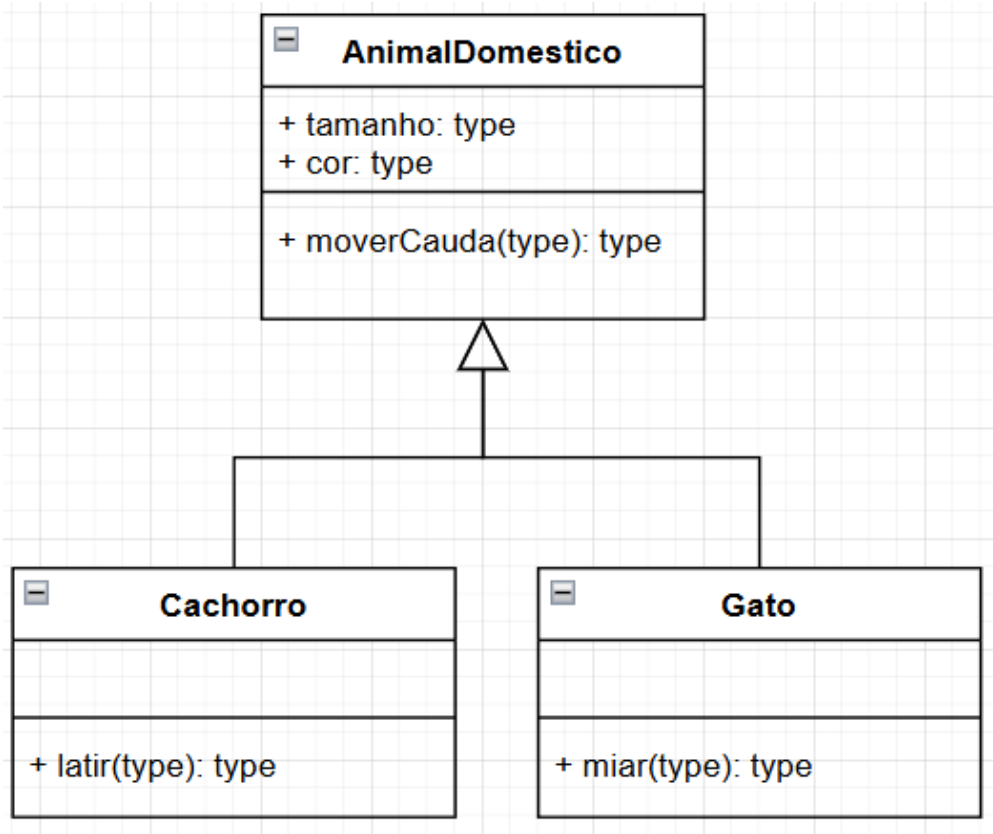
app.diagrams.net (draw.io)

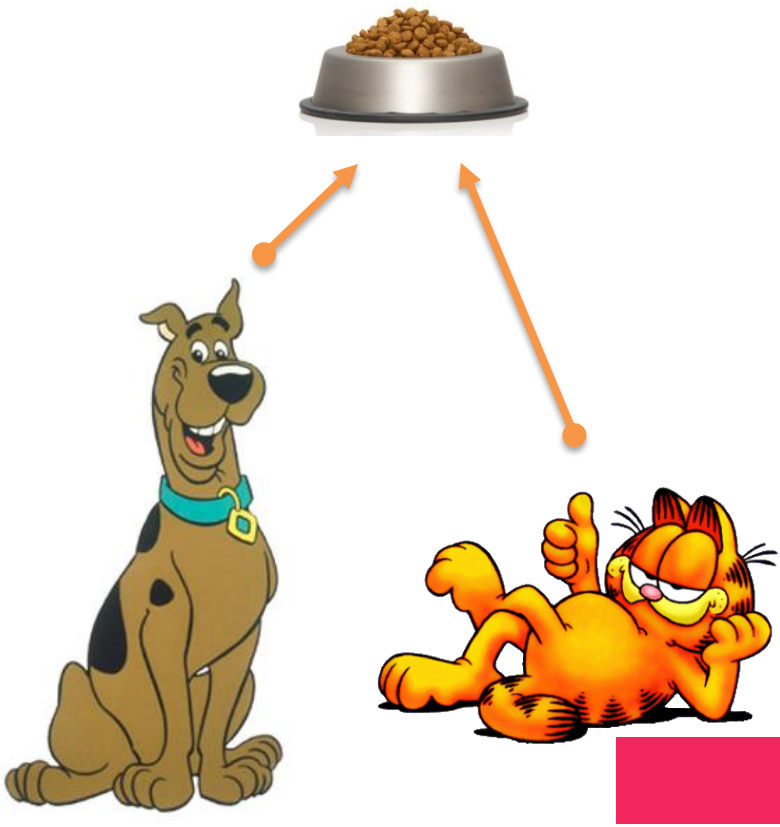


Herança

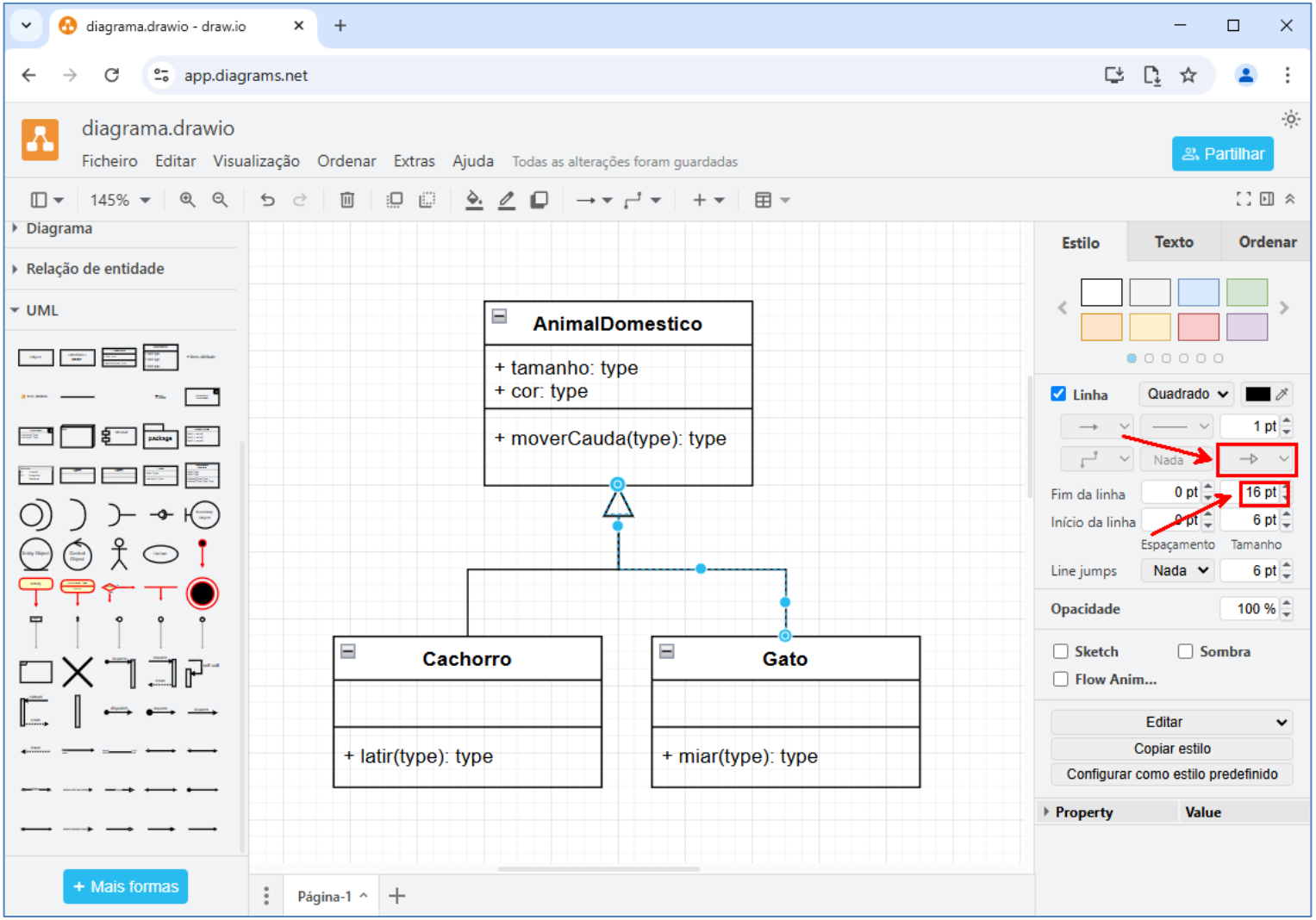
Em orientação a objeto podemos criar uma classe base e, a partir desta classe, criar subclasses relacionadas. As subclasses **herdarão** todos os atributos e métodos da classe base e poderão ter seus próprios atributos e métodos.

Exemplo:





app.diagrams.net (draw.io)



Polimorfismo

Polimorfismo significa ter ***muitas formas***, que significa um único nome representando um código diferente, selecionado por algum mecanismo automático. “Um nome, vários comportamentos”.

O Polimorfismo não é um pensamento novo para nós. Ele está contido em nosso dia a dia, principalmente na linguagem. Veja os exemplos:

1. Ontem sai para **dançar** com uns amigos, mas acabamos **dançando** porque não conseguimos encontrar um lugar que nos agradasse.

2. José **cantou** a noite inteira no Karaokê e João **cantou** a noite inteira a namorada de José.

Polimorfismo

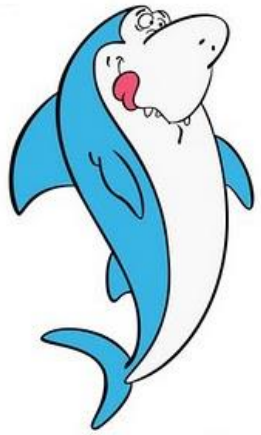
Pensando mais em objetos e funcionalidades, pense agora no termo ***abrir***, por exemplo. Você pode abrir uma porta, uma caixa, uma janela e uma conta bancária.

A palavra abrir pode ser aplicada a muitos objetos do mundo real sendo que cada objeto interpreta ***abrir*** a sua própria maneira.

Porém, você pode simplesmente dizer ***abrir***, para descrever a ação.

Os principais tipos de **polimorfismo** são:

- Sobrescrita
- Sobrecarga



Sobrescrita

Métodos com o mesmo nome, mas com funcionalidades diferentes. Exemplo:

Um objeto da classe **Carro**:

- o que ocorre quando se pressiona o pedal do acelerador enquanto se dirige cada um dos carros (básico e luxuoso)?

- o pedal do acelerador tem a capacidade de atuar de forma diferente, apesar de parecer o mesmo em todas as situações.

Cada objeto da família pode ter métodos com o mesmo nome, mas com comportamento diferente.

Sobrecarga

Ocorre quando existem dois métodos (ou mais) com mesmo nome, na mesma classe com assinaturas diferentes.

O método será escolhido de acordo com o número de parâmetros, tipo ou valor de retorno esperado.

Exemplo:

```
exibir(texto)                exibir("Olá Mundo!")
exibir(texto, número)        exibir("Astrogildo",33)
```

Teste seus conhecimentos

Utilizando suas próprias palavras, responda:

1. Explique Orientação à Objetos.
2. Explique o que é objeto.
3. Explique o que é classe.
4. Explique o que é abstração.
5. Explique o que é herança.
6. Explique o que é encapsulamento.
7. Explique o que é polimorfismo.
8. Quais os principais tipos de polimorfismo? Explique cada um deles.

FIAP

Praticando…

Praticando

- Inicie um novo projeto **Java**, crie os pacotes **br.com.fiap** e dentro deste pacote crie uma nova classe chamado **Questionario**.

- Dentro desta classe crie o método **main**.

- Dentro do método **main**, exiba no console com o método **println()** cada uma das questões e sua respectiva resposta (utilize vários métodos **println** para pular de linha).

Dica: Se você utilizar o método **println** sem colocar nenhuma mensagem, ele apenas vai pular de linha no console.

Referências

Java como programar. Paul Deitel e Harvey Deitel. Pearson, 2011.

Java 8 – Ensino Didático : Desenvolvimento e Implementação de Aplicações. Sérgio Furgeri. Editora Érica, 2015.

Até breve!