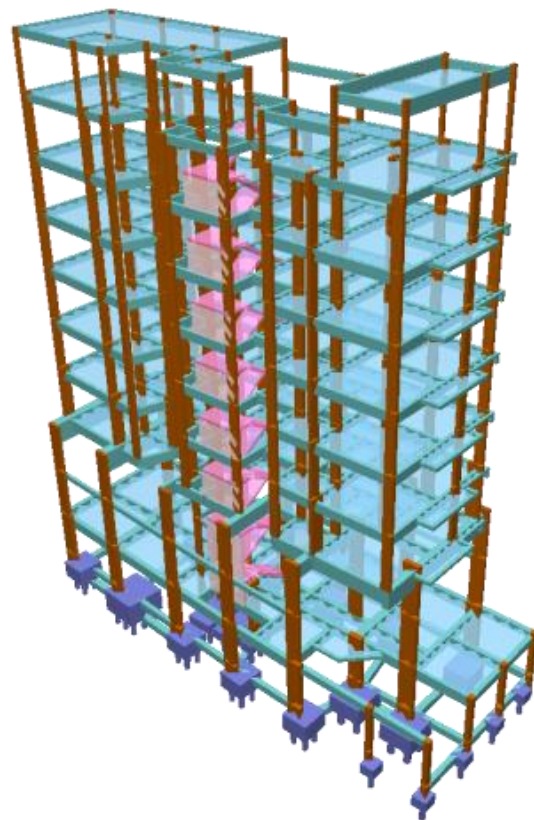


Projetos Orientados a Objetos

Engenharia de software

- Sistemas requerem abordagem multidisciplinar e devem ser projetados para durarem muitos anos em um ambiente dinâmico.
- Podemos identificar três fases nos paradigmas de desenvolvimento.
 - Definição: Determina viabilidade, requisitos do software, especifica e projeta o sistema.
 - Desenvolvimento: Implementação, integração e instalação.
 - Operação: manutenção, correção e evolução.

Modelagem e Programação



Modelos

Modelagem e Programação




Programas

Orientação a Objetos

Orientação a Objetos

- A Orientação a Objetos, é uma estratégia de desenvolvimento de software que organiza o software como uma coleção de objetos que contém tanto a estrutura dos dados quanto o comportamento deles.
- A Orientação a Objetos tem, como principal característica, a forma natural de tratar a realidade, pois considera que o mundo real é formado por objetos.
- Orientação a Objetos despende maior esforço para a fase de análise, dando ênfase às estruturas de dados antes das funções.
- Benefícios como:
 - Reutilização do código (componentes),
 - Confiabilidade (objetos encapsulados)
 - Aumento de produtividade

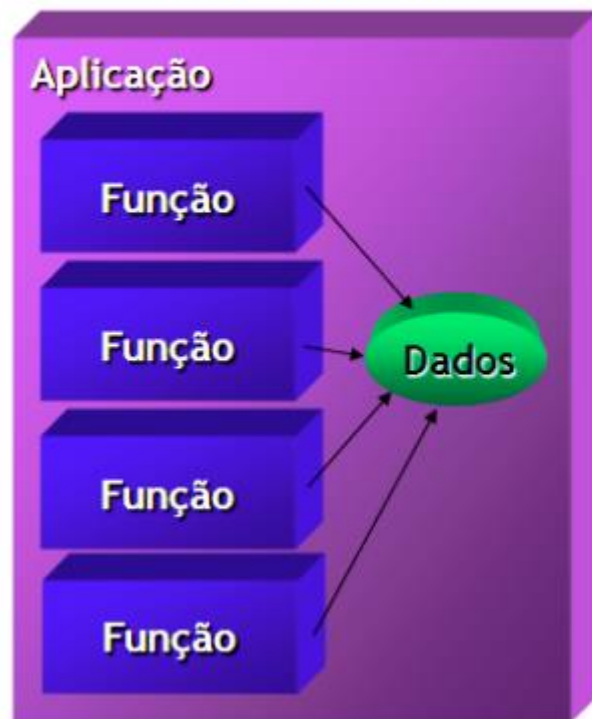


Uma habilidade crucial no desenvolvimento orientado a objetos é atribuir, responsabilidades aos objetos de software.

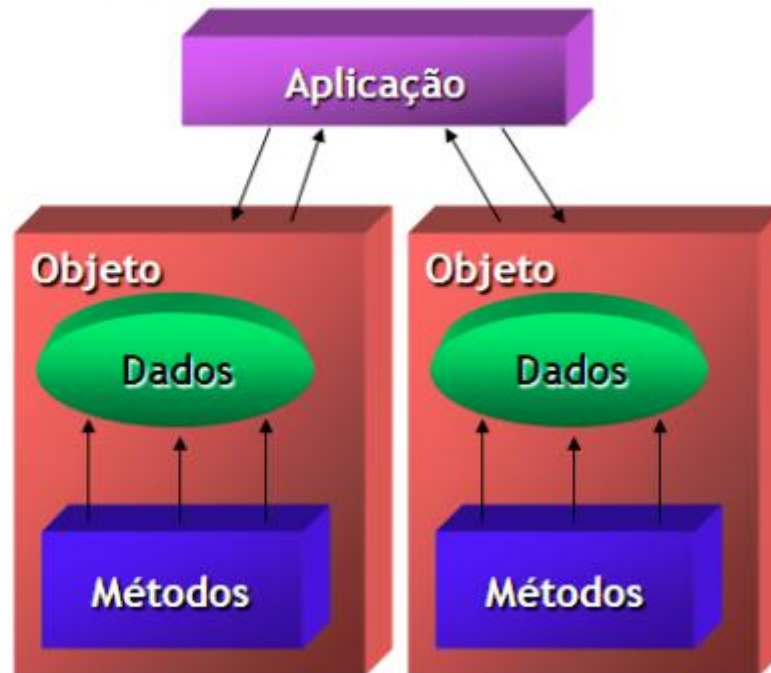
Leandro Rezende

Orientação a Objetos

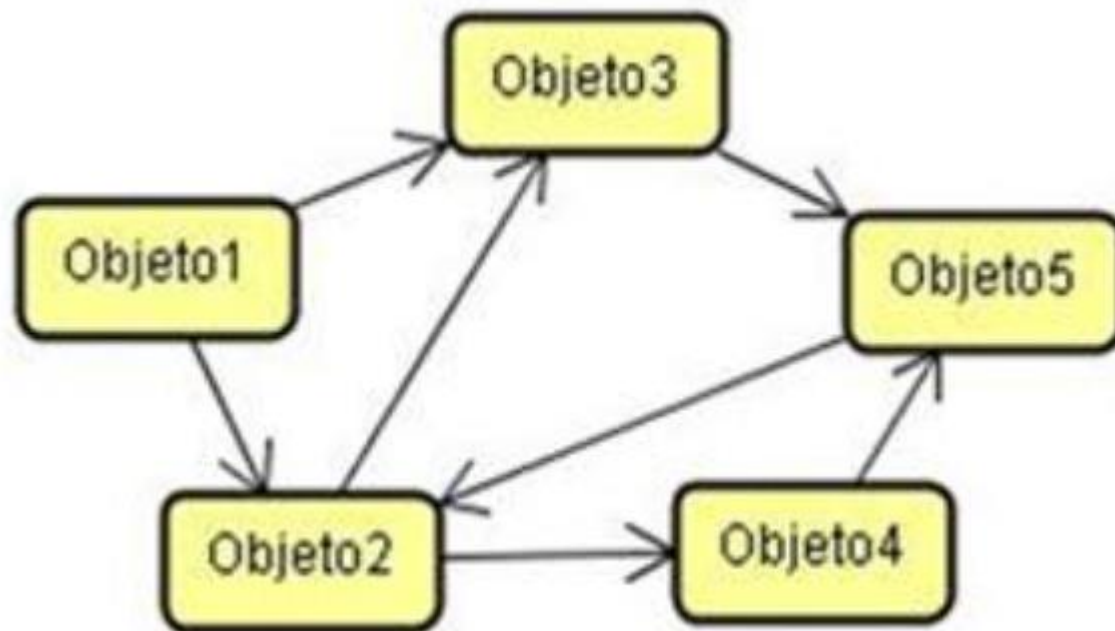
Estruturada



Orientação a Objetos




Orientação a Objetos



Orientação a Objetos






Abstração é a habilidade de se concentrar nos aspectos essenciais do sistema, ou um contexto qualquer, ignorando o que é supérfluo

Projeto

Análise e Projeto

- Análise: É um termo de significado amplo, melhor qualificado como análise de requisitos (investigação dos requisitos) ou análise orientada a objetos (investigação dos objetos do domínio).
- Projeto: É a solução conceitual que satisfaça os requisitos. Pode ou não ser implementado.



**“Faça a coisa certa
(análise).
Faça certo a coisa
(projeto).”**

O que é UML

- UML (Linguagem de Modelagem Unificada) é uma linguagem visual para especificar, construir e documentar os artefatos dos sistemas.
- Modelar sistemas (não apenas de software) usando os conceitos da orientação a objetos.
- Estabelecer uma união fazendo com que métodos conceituais sejam também executáveis.
- Criar uma linguagem de modelagem usável tanto pelo homem quanto pela máquina.

Os objetivos da UML

- Modelar sistemas (não apenas de software) usando os conceitos da orientação a objetos;
- Estabelecer uma união fazendo com que métodos conceituais sejam também executáveis;
- Criar uma linguagem de modelagem usável tanto pelo homem quanto pela máquina.
- É totalmente baseada em conceitos e padrões extensivamente testados provenientes das metodologias existentes anteriormente, e também é muito bem documentada com toda a especificação da semântica da linguagem representada em meta-modelo.

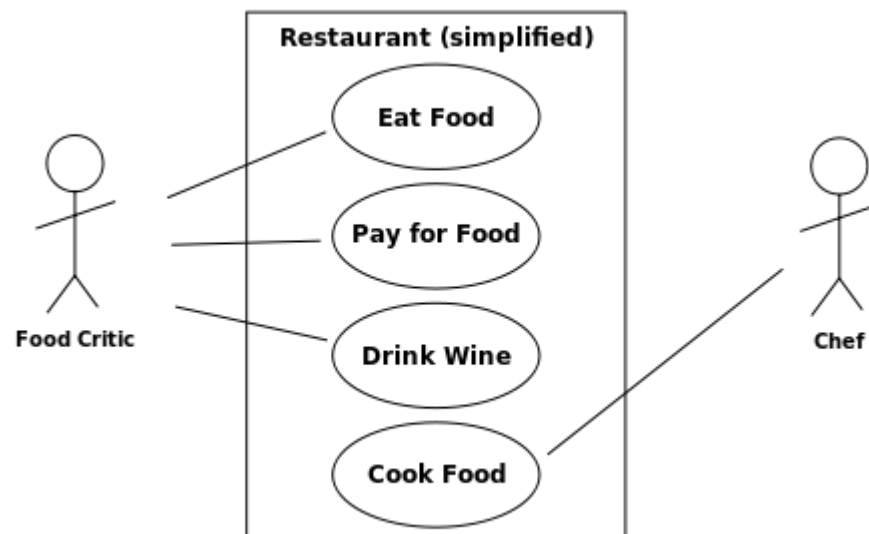
Como utilizar a UML

- Visões mostram diferentes aspectos do sistema que está sendo modelado.
- A visão é uma abstração consistindo em uma série de diagramas.
- Definindo um número de visões, cada uma mostrará aspectos particulares do sistema, dando enfoque a ângulos e níveis de abstrações diferentes e uma figura completa do sistema poderá ser construída.

Visões da UML

Visão de use-case

- Descreve a funcionalidade do sistema desempenhada pelos atores externos do sistema (usuários). A visão use-case é central, já que seu conteúdo é base do desenvolvimento das outras visões do sistema.



Visões da UML

Visão de use-case

- Descreve como a funcionalidade do sistema será implementada. A visão lógica observa e estuda o sistema internamente.
- Ela descreve e especifica a estrutura estática do sistema (classes, objetos, e relacionamentos) e as colaborações dinâmicas quando os objetos enviarem mensagens uns para os outros para realizarem as funções do sistema.
- Propriedades como persistência e concorrência são definidas nesta fase, bem como as interfaces e as estruturas de classes.

Visões da UML

Visão de lógica

- Ela descreve e especifica a estrutura estática do sistema (classes, objetos, e relacionamentos) e as colaborações dinâmicas
- A estrutura estática é descrita pelos diagramas de classes e objetos.
- A modelagem dinâmica é descrita pelos diagramas de estado, sequência, colaboração e atividade

Visões da UML

Visão de componentes

- É uma descrição da implementação dos módulos e suas dependências.
- É executado por desenvolvedores, e consiste nos componentes dos diagramas. (PACOTES)

Visões da UML

Visão de concorrência

- Trata a divisão do sistema em processos e processadores.
- Uma vez dividido o sistema em linhas de execução de processos concorrentes (threads), esta visão de concorrência deverá mostrar como se dá a comunicação e a concorrência destas threads.
- Diagramas dinâmicos: diagramas de estado, sequência, colaboração e atividade
- Diagramas de implementação: que são os diagramas de componente e execução.

Visões da UML

Visão de organização

- Mostra a organização física do sistema, os computadores, os periféricos e como eles se conectam entre si.
- Esta visão será executada pelos desenvolvedores, integradores e testadores, e será representada pelo diagrama de execução.

Partes da UML

- **Modelos de Elementos:** Os conceitos usados nos diagramas são modelos de elementos que representam definições comuns da orientação a objetos como as classes, objetos, mensagem, relacionamentos entre classes incluindo associações, dependências e heranças.
- **Mecanismos Gerais:** Os mecanismos gerais provém comentários suplementares, informações, ou semântica sobre os elementos que compõem os modelos; eles provém também mecanismos de extensão para adaptar ou estender a UML para um método/processo, organização ou usuário específico.
- **Diagramas:** Os diagramas são os gráficos que descrevem o conteúdo em uma visão. UML possui nove tipos de diagramas que são usados em combinação para prover todas as visões do sistema.

Modelos de Elementos

Objetos

- Representação computacional de algo do mundo real
 - Concreto
 - Abstrato
- Abstração
 - Transformar aquilo que observamos realidade para a virtualidade

Modelos de Elementos

Objetos

- Concretos
 - Cão
 - Moto
 - Casa
- Abstratos
 - Música
 - Transação Bancária
- Modelo
 - Características + Comportamento

Modelos de Elementos

Objetos

- Estado
 - Atributos (Características)
- Operações
 - Métodos (Comportamentos)
- Identidade
 - Dois objetos com estado e operações precisamente idênticos não são iguais
- Operações podem mudar os valores dos atributos assim mudando o estado de um objeto.

Modelos de Elementos

Métodos e Atributos



| Atributos | Métodos |
|-----------------|-----------|
| Raça: Poodle | Latir |
| Nome: Doki | Comer |
| Peso: 7kg | Dormir |
| | |
| Potência: 500cc | Acelerar |
| Modelo: Honda | Frear |
| Ano: 1998 | Abastecer |

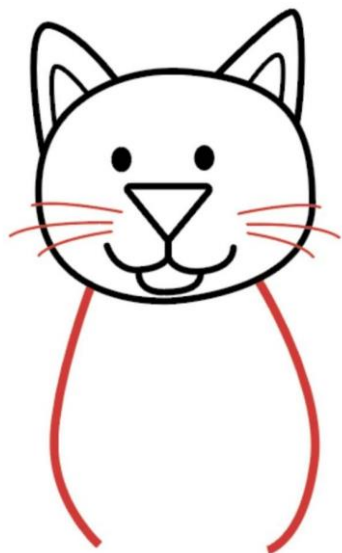
Modelos de Elementos

Classe

- Conjunto de objetos:
 - Características semelhantes
 - Comportamento comum
 - Interação com outros objetos
- Uma classe é a forma para criação de objetos
- Objetos são representações concretas (instâncias) de uma classe

Modelos de Elementos

Classe



Classe Gato

New

New

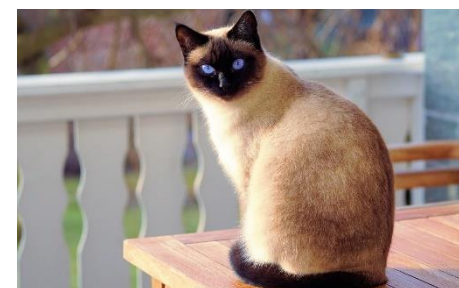
New



Raça: Savannah
Nome: Gatuno
Peso: 2,5 quilos
Idade: 2 anos



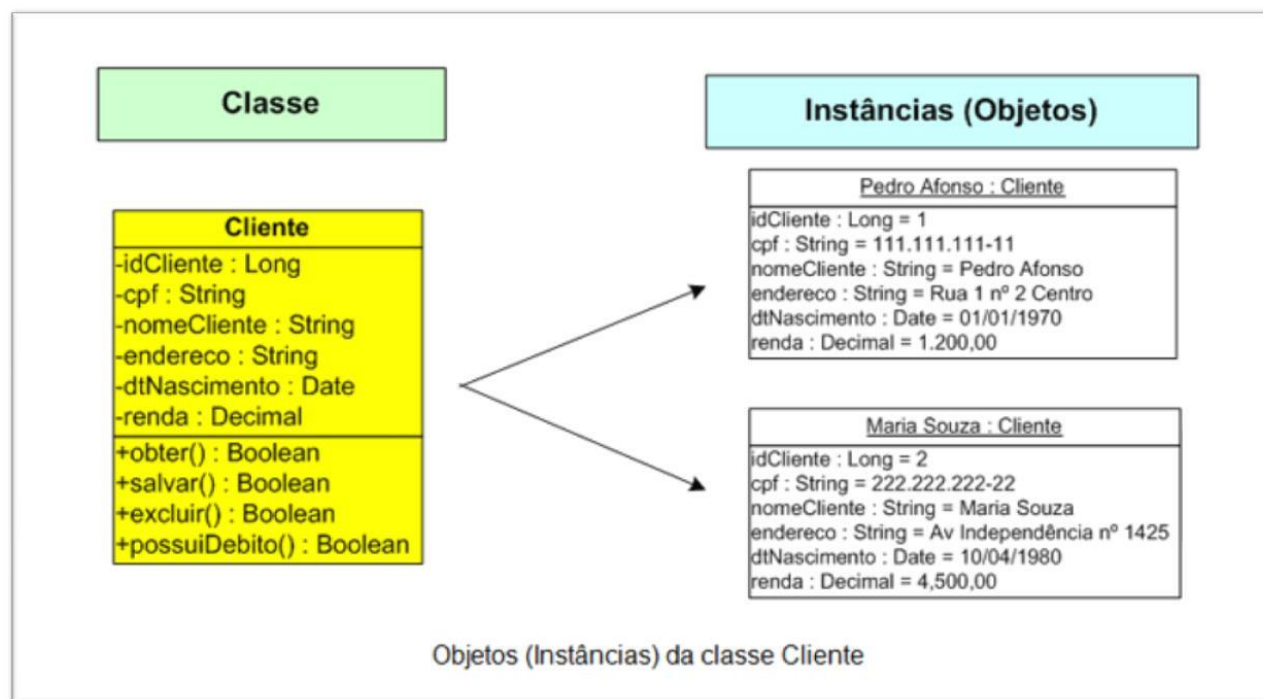
Raça: Maine Moon
Nome: Listrado
Peso: 3 quilos
Idade: 5 anos



Raça: Siamês
Nome: Bichano
Peso: 4 quilos
Idade: 3 anos

Modelos de Elementos

Classe - Objeto



Modelos de Elementos

Encapsulamento

- Um objeto, em um programa, “encapsula” todo o seu estado e o comportamento;
- Os dados e as operações são agrupados e a sua implementação é escondida, protegida dos usuários;

Modelos de Elementos

Visibilidade

- Private

- Somente a classe tem acesso Não é transmitido por herança

~ Default ou Friendly

- Acesso a classe inteira
 - Visível para as classes do mesmo pacote
- Só é transmitido por herança em classes do mesmo pacote

Protected




















- Visível em toda a classe
- Visível em todas as classes de um pacote
- Transmitido por herança

+ Public

- Visível irrestritamente

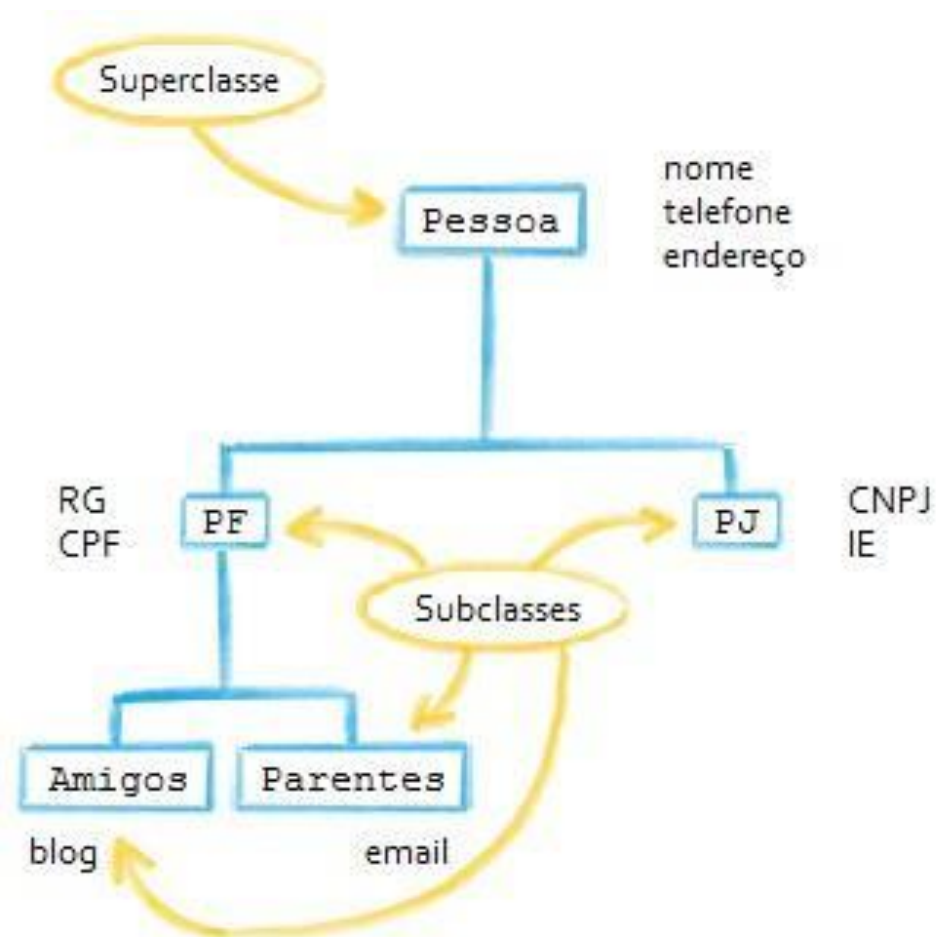
Modelos de Elementos

Visibilidade

| Visibilidade | <u>public</u> | <u>protected</u> | default | <u>private</u> |
|---|---|---|---|---|
| A partir da mesma classe |  |  |  |  |
| Qualquer classe no mesmo pacote |  |  |  |  |
| Qualquer classe filha no mesmo pacote |  |  |  |  |
| Qualquer classe filha em pacote diferente |  |  |  |  |
| Qualquer classe em pacote diferente |  |  |  |  |

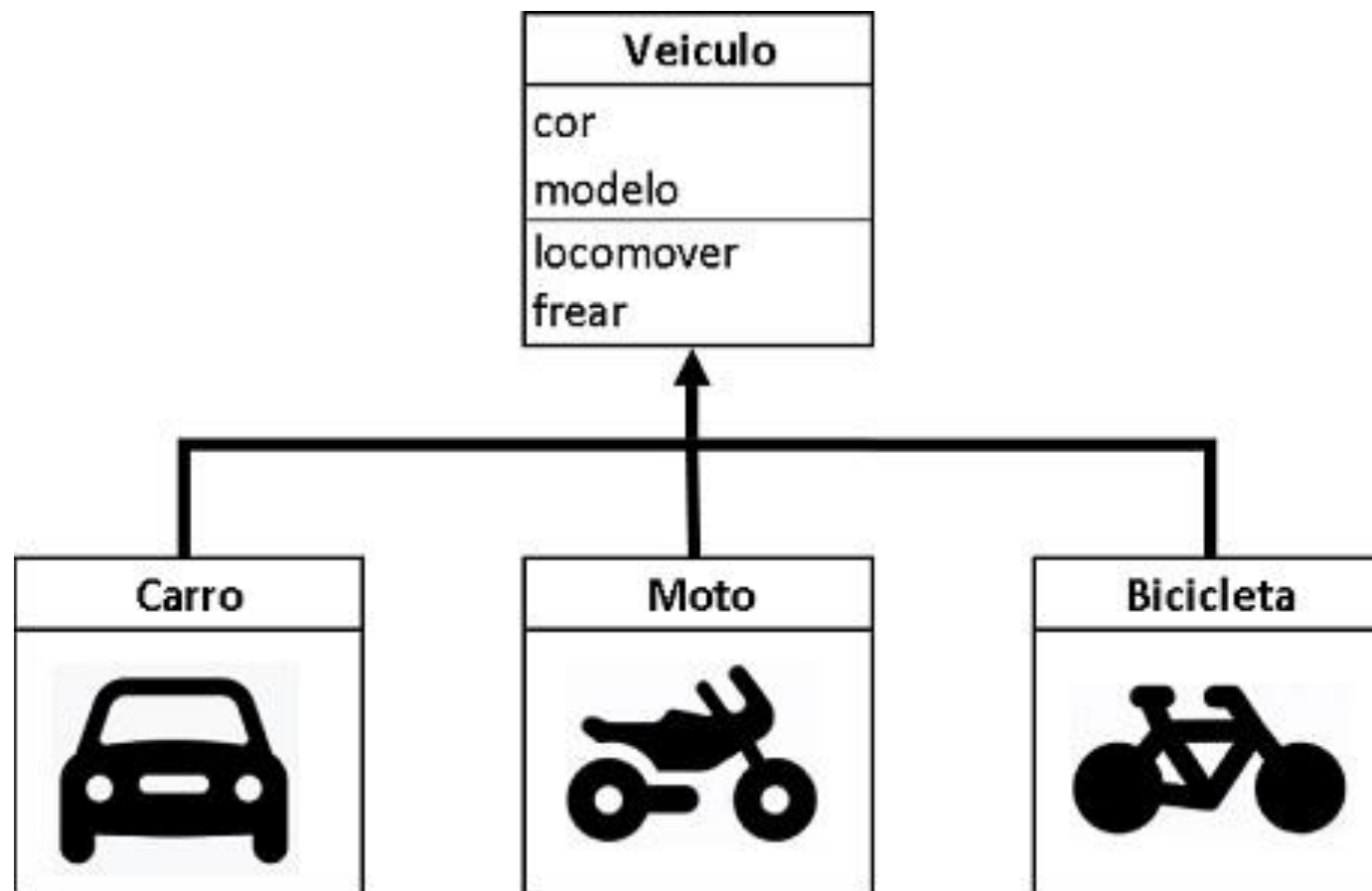
Modelos de Elementos

Herança



Modelos de Elementos

Herança



Modelos de Elementos

Polimorfismo



Modelos de Elementos

Polimorfismo





#2022
Realizar

