



***Faculdade de
Tecnologia SENAI
“Roberto Mange”***



***Engenharia de Software
Modelagem - 1***

Objetivos

- Entender a importância do uso de modelos para o desenvolvimento de software;
- Apresentar uma visão geral sobre a notação gráfica UML
- Conhecer os principais diagramas UML



Graduação Tecnológica
Engenharia de Software

Motivação

Existe uma lacuna entre os seguintes mundos:

- Requisitos: o que o sistema faz (abstração mais alta)
- Código: como o sistema faz isso (abstração mais baixa)



Graduação Tecnológica
Engenharia de Software

Modelagem

Objetivo: preencher essa "lacuna"

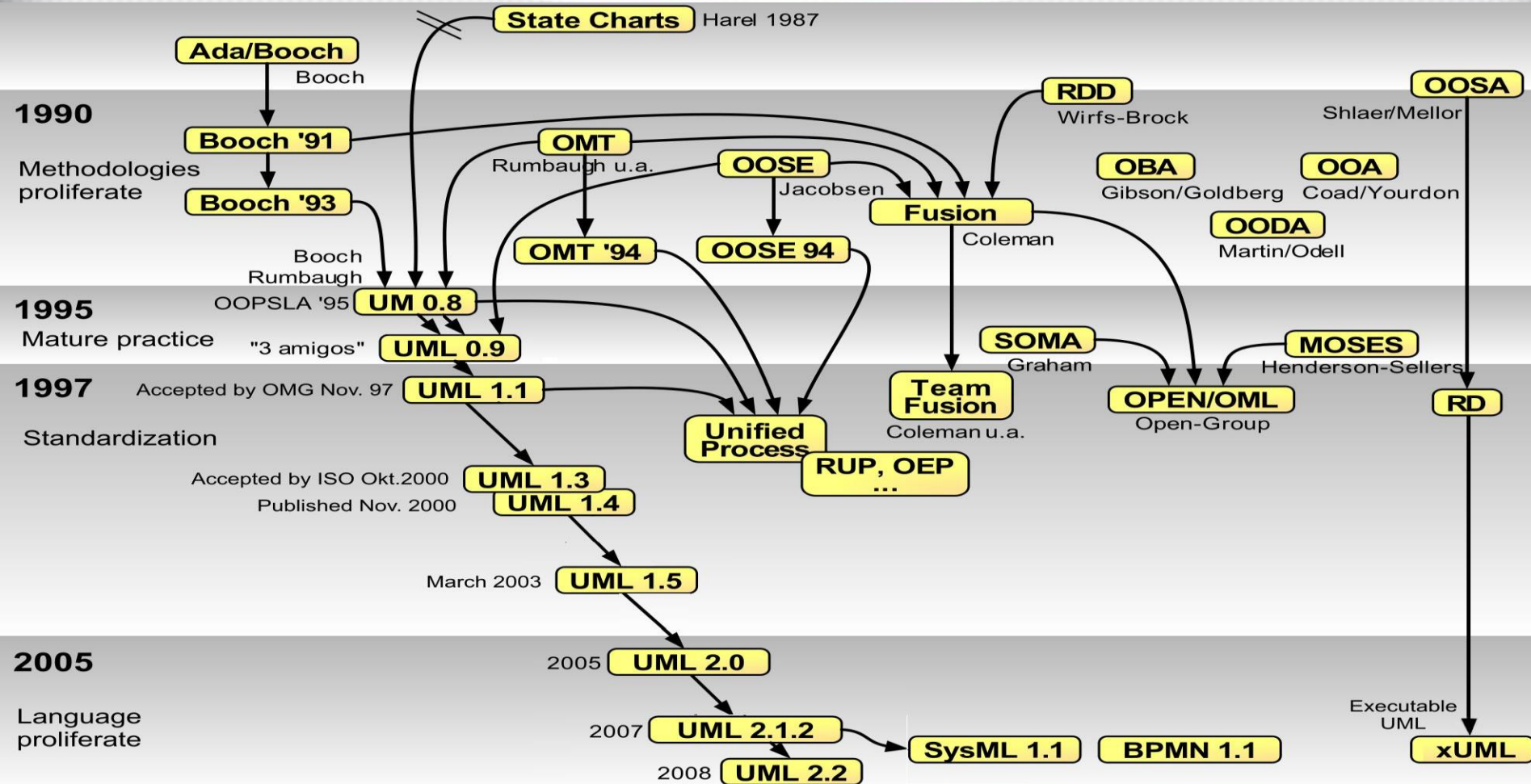
Via uma notação com um nível de abstração intermediário

Documentar uma solução para o problema definido pelos requisitos para os desenvolvedores e em alguns casos para o usuário (stakeholders)

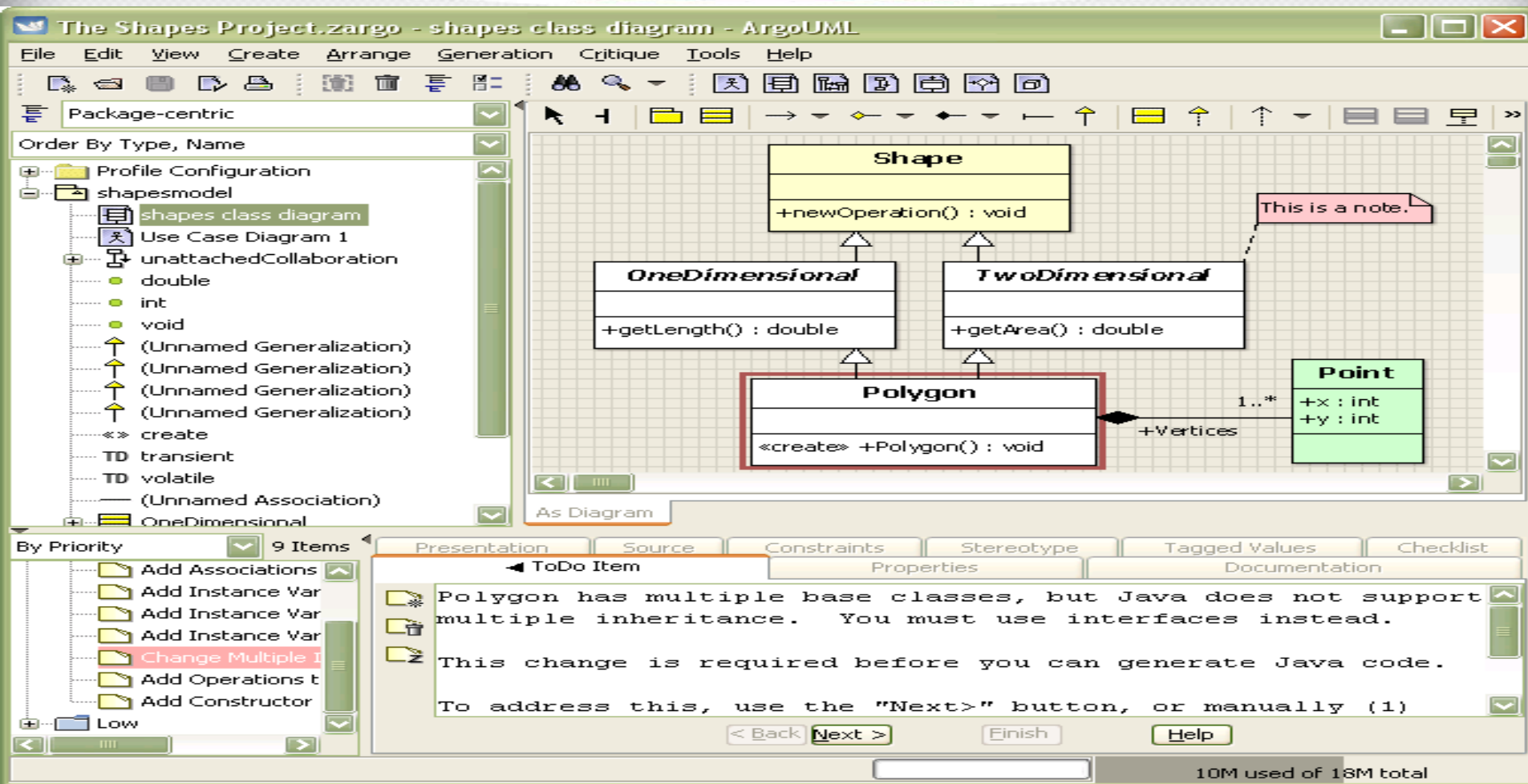


Graduação Tecnológica
Engenharia de Software

UML



Ferramentas CASE



Usos da UML - Blueprint

- Conjunto de modelos após os requisitos
- Documentação do sistema usando UML
- Modelos são repassados para ser codificados
- Mais usado em metodologia WATERFALL E UP
- Não é comum seu uso
- Uso massivo de ferramentas case



Graduação Tecnológica
Engenharia de Software

Usos da UML – MDD

model driven development

- Criada com a intenção de gerar códigos a partir de modelos
- UML extendida com diversos outros tipos de diagramas criados
- Não houve até agora um uso significativo, mas com o desenvolvimento da IA....



Graduação Tecnológica
Engenharia de Software

Usos da UML – Sketches

- Usada para construir diagramas leves e particionados de um sistema
- Uso mais comum nos métodos ágeis
- Usado para conversar sobre uma parte do código ou do projeto e documentar uma parte do código ou do projeto
- Uso de apenas um subconjunto dos diagramas UML



Graduação Tecnológica
Engenharia de Software

Usos da UML – Sketches

- Não se tem o objetivo de gerar modelos complexos ou detalhados
- Também não se tem a intenção de gerar códigos automáticos com esses modelos, mas com a IA...
- Dispensa-se o uso de ferramentas CASE caras



Graduação Tecnológica
Engenharia de Software

Usos da UML – Sketches

Engenharia Avante

- > Para discutir e analisar alternativas de design antes que exista qualquer código
- > Objetivo é validar a proposta de projeto antes de começar a codificar

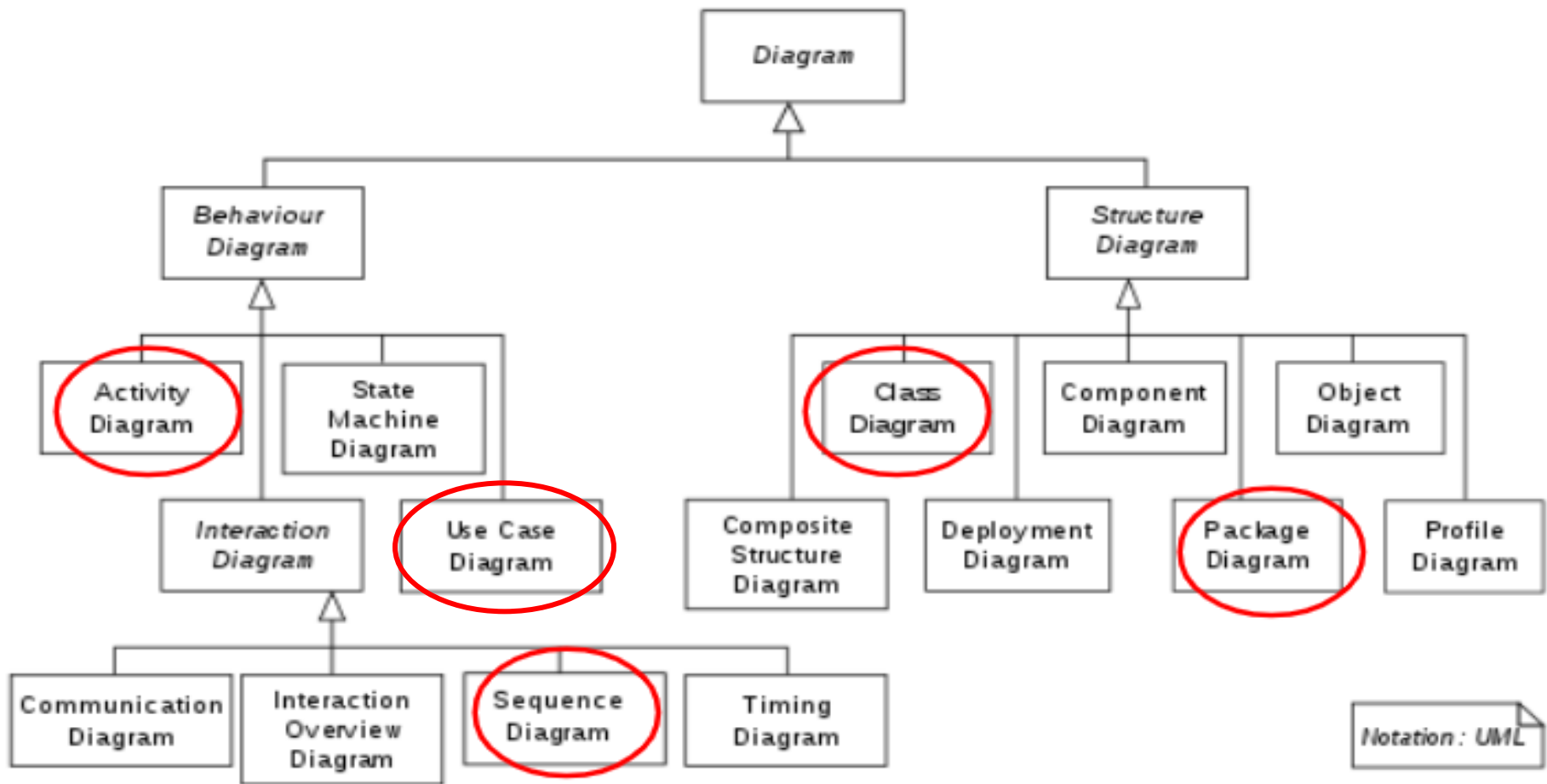
Engenharia Reversa

- > Para analisar e discutir uma funcionalidade já implementada no código fonte
- > Pode ser usada para explicar como uma funcionalidade está implementada



Graduação Tecnológica
Engenharia de Software

Tipos de Diagramas



Diagramas Estáticos ou estruturais

Estáticos ou Estruturais

Modelam a estrutura e organização de um sistema

Incluem informações sobre classes, atributos, métodos, pacotes, etc

Ex: Diagramas de Classes e Diagramas de Pacotes

Lidam apenas com informações que estão disponíveis



Graduação Tecnológica
Engenharia de Software

Diagramas Dinâmicos ou comportamentais

Dinâmicos ou Comportamentais

Modelam o comportamento de um programa

Incluem o que pode acontecer durante a execução, quais métodos são de fato executados, etc

Ex: Diagramas de Sequência e Diagramas de Atividades

Fornecem uma visão de tempo de execução

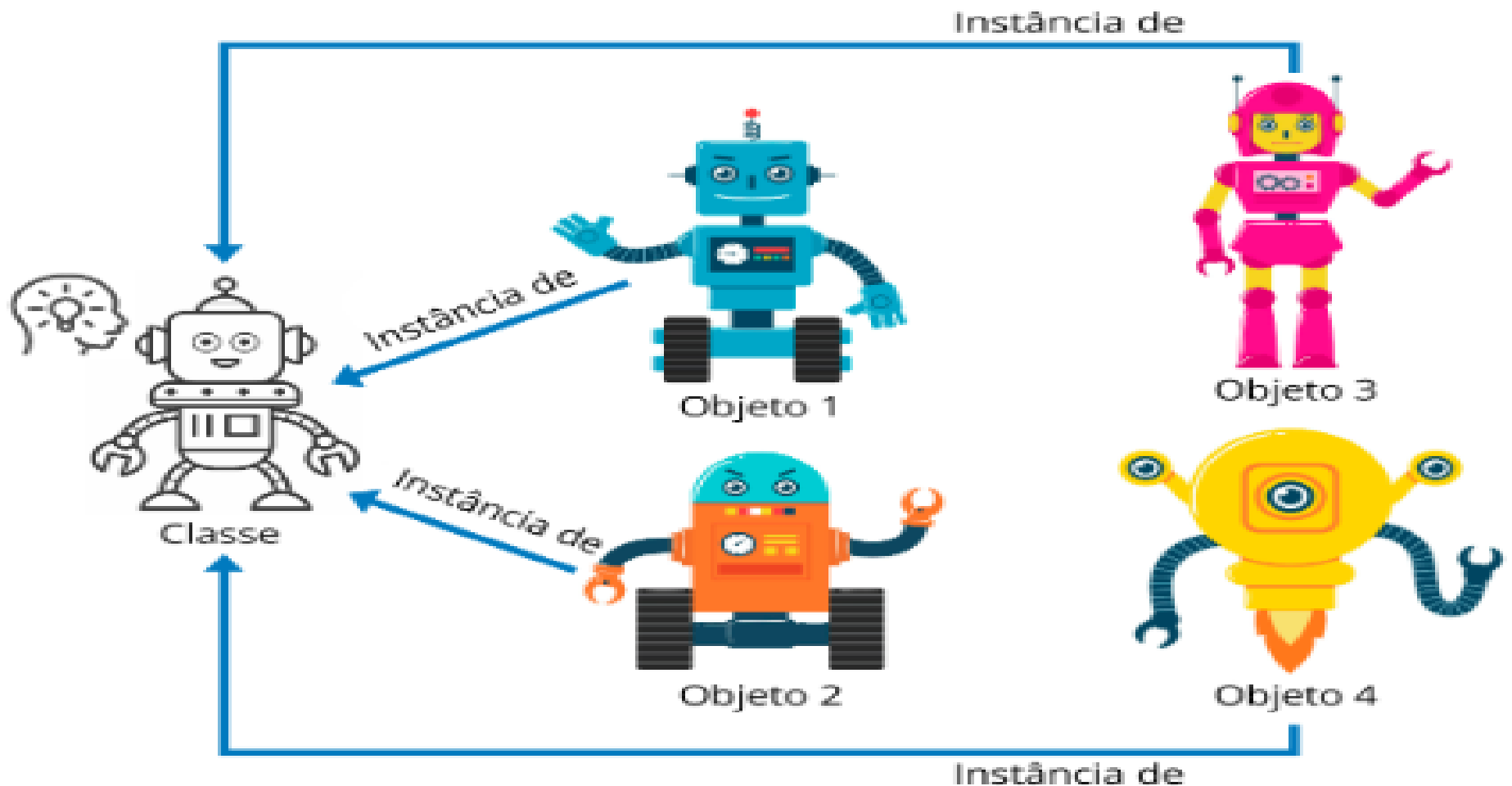


Graduação Tecnológica
Engenharia de Software

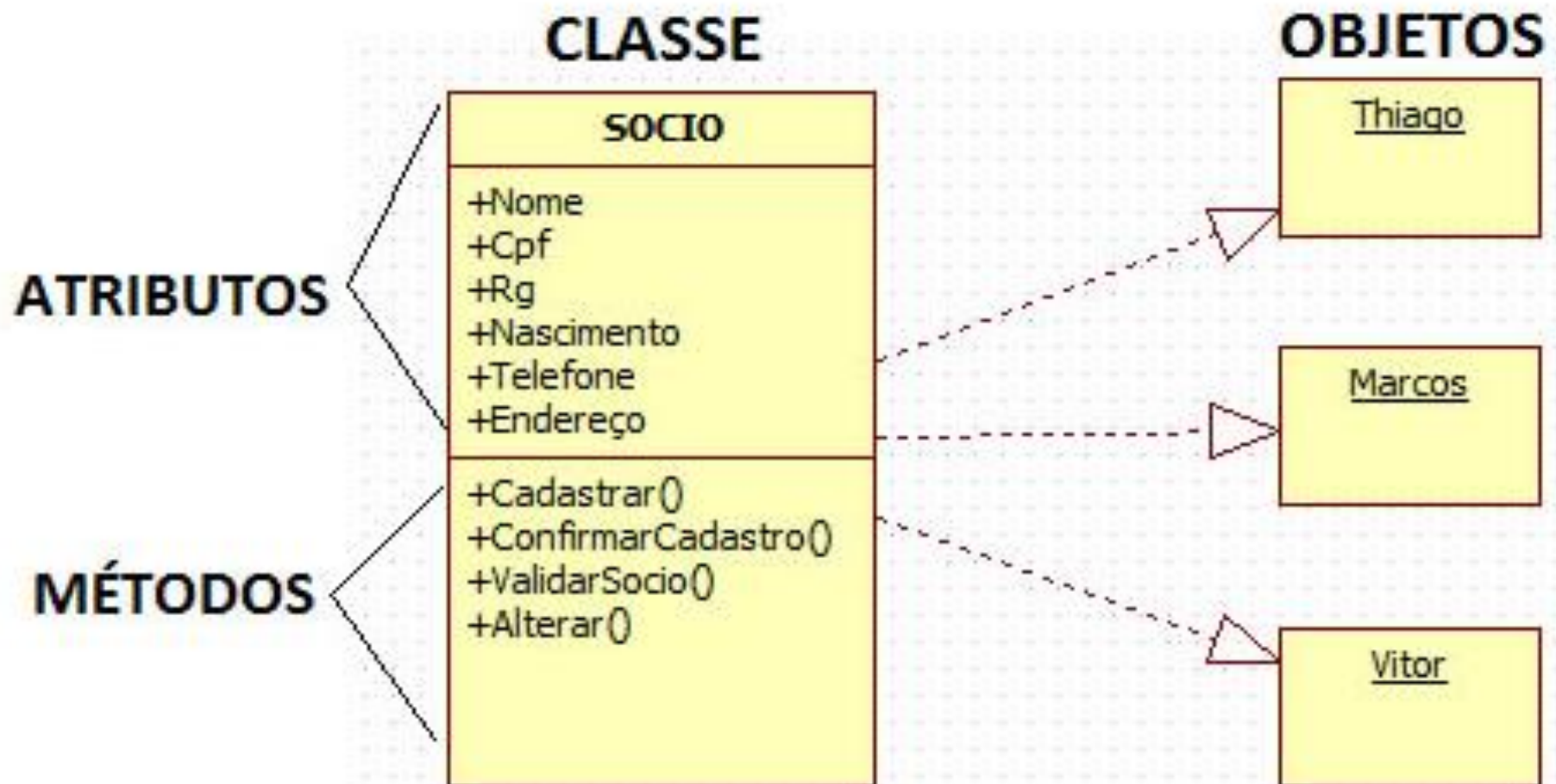
Básico de Orientação a Objetos

- **Objeto** é uma entidade que pode ser identificada univocamente através de suas **propriedades e comportamentos**.
- **Classe** é um grupo de **objetos similares** que compartilham atributos e comportamentos semelhantes.
- A **orientação a objetos** permite **modelar** o sistema utilizando os conceitos do **mundo real**.

Classe x Objetos



Atributos e Métodos

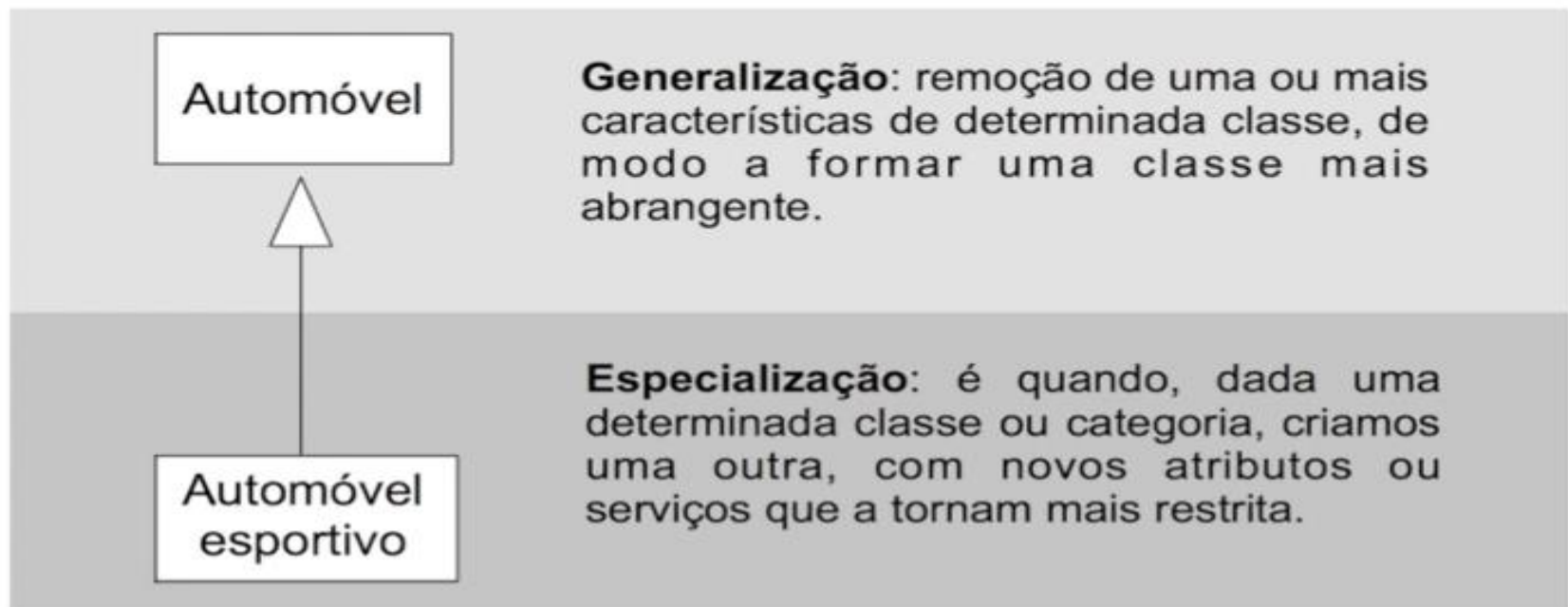


Pilares



Abstração/Herança

Compartilhamento de atributos e operações entre classes com base em um relacionamento hierárquico.



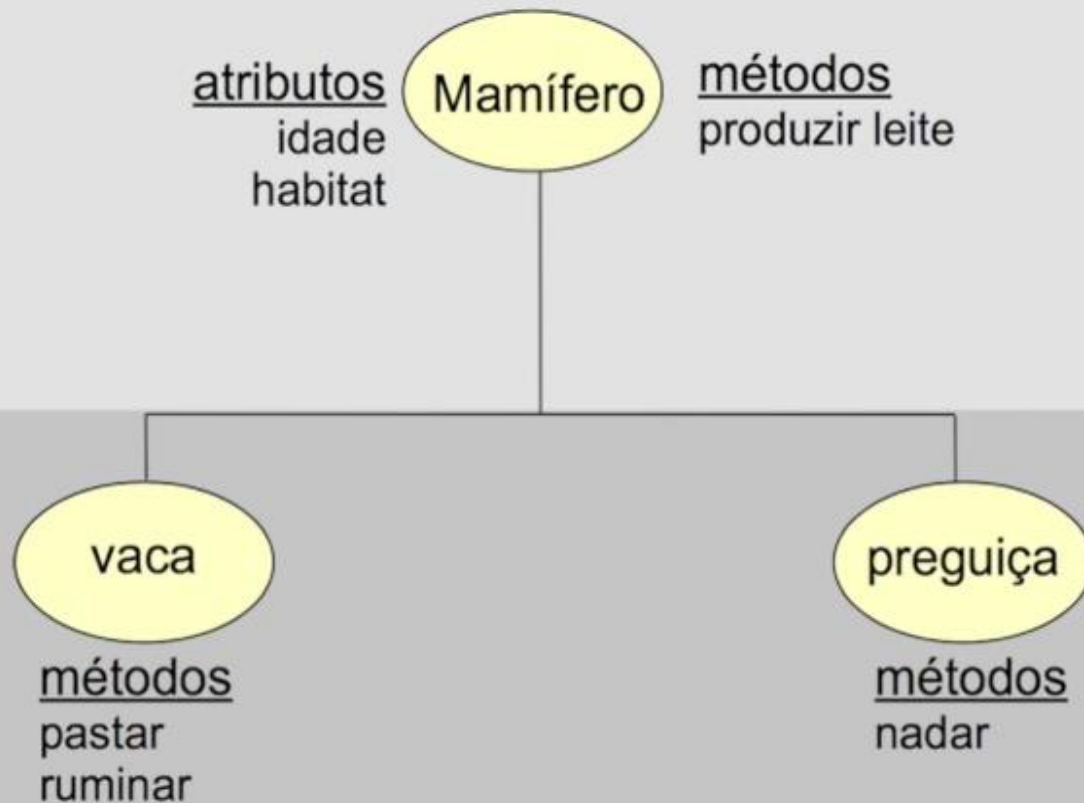
Vantagens da Abstração/Herança

- Cada **subclasse incorpora** (herda) todas as **características** de sua **superclasse** e **acrescenta** suas próprias e **específicas**.
- Vantagens
 - Simplificação da codificação
 - Alterações em classes pai são propagadas para as filhas
 - Reuso

Exemplo de Abstração/Herança

Generalização

Superclasse



Especialização

Subclasse

Exemplo de Abstração/Herança Python

```
class Animal:
    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade

    def descrever(self):
        print(f"Nome: {self.nome}, Idade: {self.idade}")

class Cachorro(Animal):
    def __init__(self, nome, idade, raca):
        super().__init__(nome, idade) # Inicializa atributos da classe pai
        self.raca = raca

    def latir(self):
        print("Au au!")

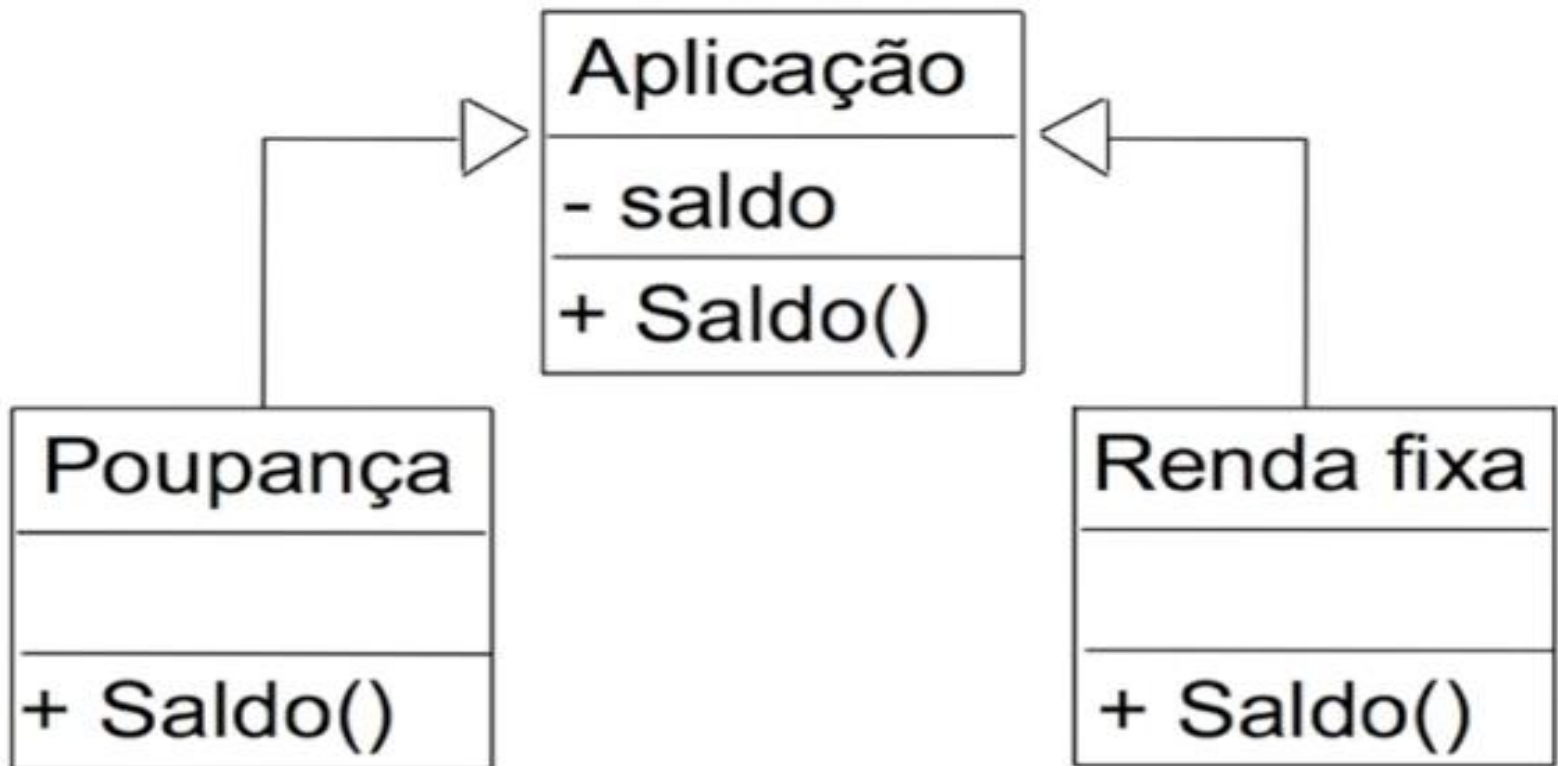
    def descrever(self):
        super().descrever() # Chama o método da classe pai
        print(f"Raça: {self.raca}")
```

```
animal = Animal("Bichinho", 5)
animal.descrever() # Saída: Nome: Bichinho, Idade: 5

cachorro = Cachorro("Rex", 3, "Labrador")
cachorro.latir() # Saída: Au au!
cachorro.descrever() # Saída: Nome: Rex, Idade: 3
                     # Raça: Labrador
```

Polimorfismo

Polimorfismo (do grego, “muitas formas”)



Polimorfismo

Em programação, o **polimorfismo** é utilizado em inúmeras situações que **ajudam na simplificação** do código e **aumentam a manutenibilidade**:

- Sobrecarga de métodos
- Sobrescrita de métodos
- Tratamento de conjuntos de objetos em contêineres.



Graduação Tecnológica
Engenharia de Software

Exemplo Polimorfismo

Deseja-se implementar uma promoção onde o desconto aplicado a cada produto é diferente:

- Livros: 10% de desconto.
- Eletrônicos: 5% de desconto.
- Roupas: 15% de desconto.



Graduação Tecnológica
Engenharia de Software

Exemplo Sem Polimorfismo

```
class Livro:
    def __init__(self, preco):
        self.preco = preco

class Eletronico:
    def __init__(self, preco):
        self.preco = preco

class Roupa:
    def __init__(self, preco):
        self.preco = preco

# Criando objetos dos produtos
livro = Livro(100)
eletronico = Eletronico(200)
roupa = Roupa(50)

# Lista de produtos
produtos = [livro, eletronico, roupa]
```

```
# Aplicando desconto sem polimorfismo
for produto in produtos:
    if isinstance(produto, Livro):
        produto.preco -= produto.preco * 0.1
    elif isinstance(produto, Eletronico):
        produto.preco -= produto.preco * 0.05
    elif isinstance(produto, Roupa):
        produto.preco -= produto.preco * 0.15

# Imprimindo os preços dos produtos com desconto
print(f"Preço do livro: {livro.preco}")
print(f"Preço do eletrônico: {eletronico.preco}")
print(f"Preço da roupa: {roupa.preco}")
```

Exemplo Com Polimorfismo

```
class Livro:
    def __init__(self, preco):
        self.preco = preco

    def calcular_desconto(self):
        self.preco -= self.preco * 0.1

class Eletronico:
    def __init__(self, preco):
        self.preco = preco

    def calcular_desconto(self):
        self.preco -= self.preco * 0.05

class Roupa:
    def __init__(self, preco):
        self.preco = preco

    def calcular_desconto(self):
        self.preco -= self.preco * 0.15

# Criando objetos dos produtos
livro = Livro(100)
eletronico = Eletronico(200)
roupa = Roupa(50)
```

```
# Lista de produtos
```

```
produtos = [livro, eletronico, roupa]
```

```
# Aplicando desconto com polimorfismo
```

```
for produto in produtos:
    produto.calcular_desconto()
```

```
# Imprimindo os preços dos produtos com desconto
```

```
print(f"Preço do livro: {livro.preco}")
```

```
print(f"Preço do eletrônico: {eletronico.preco}")
```

```
print(f"Preço da roupa: {roupa.preco}")
```

Encapsulamento

Imagine uma caixa com um cadeado: você só pode acessar o conteúdo da caixa através da fechadura, usando a chave correta.

No encapsulamento, a caixa representa uma classe e seu conteúdo são os dados (atributos) e as operações (métodos) que compõem a classe. A fechadura representa os métodos de acesso, que são portas de entrada controladas para manipular os dados internos.



Encapsulamento

Visibilidade

A visibilidade dos membros (atributos e métodos) de uma classe define quem pode acessá-los:

- Público (***Public* +**): Visível para todos, incluindo código externo à classe;
- Protegido (***Protected* #**): Visível para métodos da própria classe e de suas subclasses e para as classes do mesmo pacote
- Privado (***Private* -**): É mais restritivo e visível somente para métodos da própria classe. No Python é utilizado o underscore antes do atributo para indicar um dado privado.

Ex: class Pessoa:

```
def __init__(self, nome, idade):  
    self._nome = nome # Atributo privado  
    self._idade = idade
```



Encapsulamento

Getters e Setters

São métodos públicos (visíveis fora da classe) que fornecem acesso controlado aos atributos privados.

Getters: Retornam o valor de um atributo privado.

Setters: Modificam o valor de um atributo privado, geralmente com validações para garantir a integridade dos dados.



Encapsulamento

Exemplo 1 (Python)

```
class Pessoa:
    def __init__(self, nome, idade):
        self._nome = nome # Atributo privado
        self._idade = idade

    def get_nome(self):
        return self._nome

    def set_nome(self, novo_nome):
        self._nome = novo_nome

    def get_idade(self):
        return self._idade

    def set_idade(self, nova_idade):
        if nova_idade > 0:
            self._idade = nova_idade
        else:
            print("Idade inválida!")
```

Criando um objeto da classe Pessoa

```
pessoa1 = Pessoa("João", 30)
```

Acessando os atributos usando os getters

```
print(pessoa1.get_nome()) # Saída: João
```

```
print(pessoa1.get_idade()) # Saída: 30
```

Modificando o atributo idade usando o setter

```
pessoa1.set_idade(32)
```

```
print(pessoa1.get_idade()) # Saída: 32
```

Tentando modificar a idade com um valor inválido

```
pessoa1.set_idade(-5) # Saída: Idade inválida!
```

Encapsulamento

Exemplo 2 (JAVA)

```
public class Pessoa {  
    private String nome;  
    private int idade;  
  
    public Pessoa(String nome, int idade) {  
        this.nome = nome;  
        this.idade = idade;  
    }  
  
    public String getNome() {  
        return nome;  
    }  
  
    public void setNome(String nome) {  
        this.nome = nome;  
    }  
  
    public int getIdade() {  
        return idade;  
    }  
  
    public void setIdade(int idade) {  
        if (idade > 0) {  
            this.idade = idade;  
        } else {  
            System.out.println("Idade inválida!");  
        }  
    }  
}
```

```
public static void main(String[] args) {  
    Pessoa pessoa1 = new Pessoa("João", 30);  
  
    // Acessando os atributos usando os getters  
    System.out.println(pessoa1.getNome()); // Saída: João  
    System.out.println(pessoa1.getIdade()); // Saída: 30  
  
    // Modificando o atributo idade usando o setter  
    pessoa1.setIdade(32);  
    System.out.println(pessoa1.getIdade()); // Saída: 32  
  
    // Tentando modificar a idade com um valor inválido  
    pessoa1.setIdade(-5); // Saída: Idade inválida!  
}
```

Encapsulamento

Exemplo 2 (JAVA)

```
public class Pessoa {  
    private String nome;  
    private int idade;  
  
    public Pessoa(String nome, int idade) {  
        this.nome = nome;  
        this.idade = idade;  
    }  
  
    public String getNome() {  
        return nome;  
    }  
  
    public void setNome(String nome) {  
        this.nome = nome;  
    }  
  
    public int getIdade() {  
        return idade;  
    }  
  
    public void setIdade(int idade) {  
        if (idade > 0) {  
            this.idade = idade;  
        } else {  
            System.out.println("Idade inválida!");  
        }  
    }  
}
```

```
public static void main(String[] args) {  
    Pessoa pessoa1 = new Pessoa("João", 30);  
  
    // Acessando os atributos usando os getters  
    System.out.println(pessoa1.getNome()); // Saída: João  
    System.out.println(pessoa1.getIdade()); // Saída: 30  
  
    // Modificando o atributo idade usando o setter  
    pessoa1.setIdade(32);  
    System.out.println(pessoa1.getIdade()); // Saída: 32  
  
    // Tentando modificar a idade com um valor inválido  
    pessoa1.setIdade(-5); // Saída: Idade inválida!  
}
```

Diagrama de casos de USO

Caso de uso especifica o **comportamento** de um **sistema** ou de **parte** de um **sistema** e é uma **descrição** de um conjunto de **sequências** de **ações**, incluindo variantes realizadas pelo sistema para **produzir** um **resultado observável** do valor de um **ator**.



Diagrama de casos de USO

Concepção	Definição			Construção			Finalização
-----------	-----------	--	--	------------	--	--	-------------

- Captura de requisitos
- **Identificação** dos **agentes externos** ao sistema
- **Identificação** de **cenários** (o que o sistema deve fazer)
- Modelo descritivo



Diagrama de casos de USO

Conceitos

- **Ator:** Representa um papel (*role*) desempenhado pelo usuário (agente externo) quando este interage com o caso de uso;
 - Tipicamente **humanos**, dispositivo de **hardware** ou um **sistema** que interaja com o sistema
 - **Interagir** == estimula/**solicita ações**/eventos do sistema e **recebe reações**
- **Caso de uso:** Representa a funcionalidade (ou parte) de um sistema.

Diagrama de casos de USO

Conceitos

Como identificar atores

- Quem usa o sistema?
- Quem inicializa o sistema?
- Quem fornece os dados?
- Quem usa as informações?



Diagrama de casos de USO

Conceitos

Como identificar Casos de Uso

- Os **casos de uso** são **interações** entre os **atores** e o **sistema**.
- Os **atores** sempre **iniciam** a **ação**.



Diagrama de casos de USO

Notações

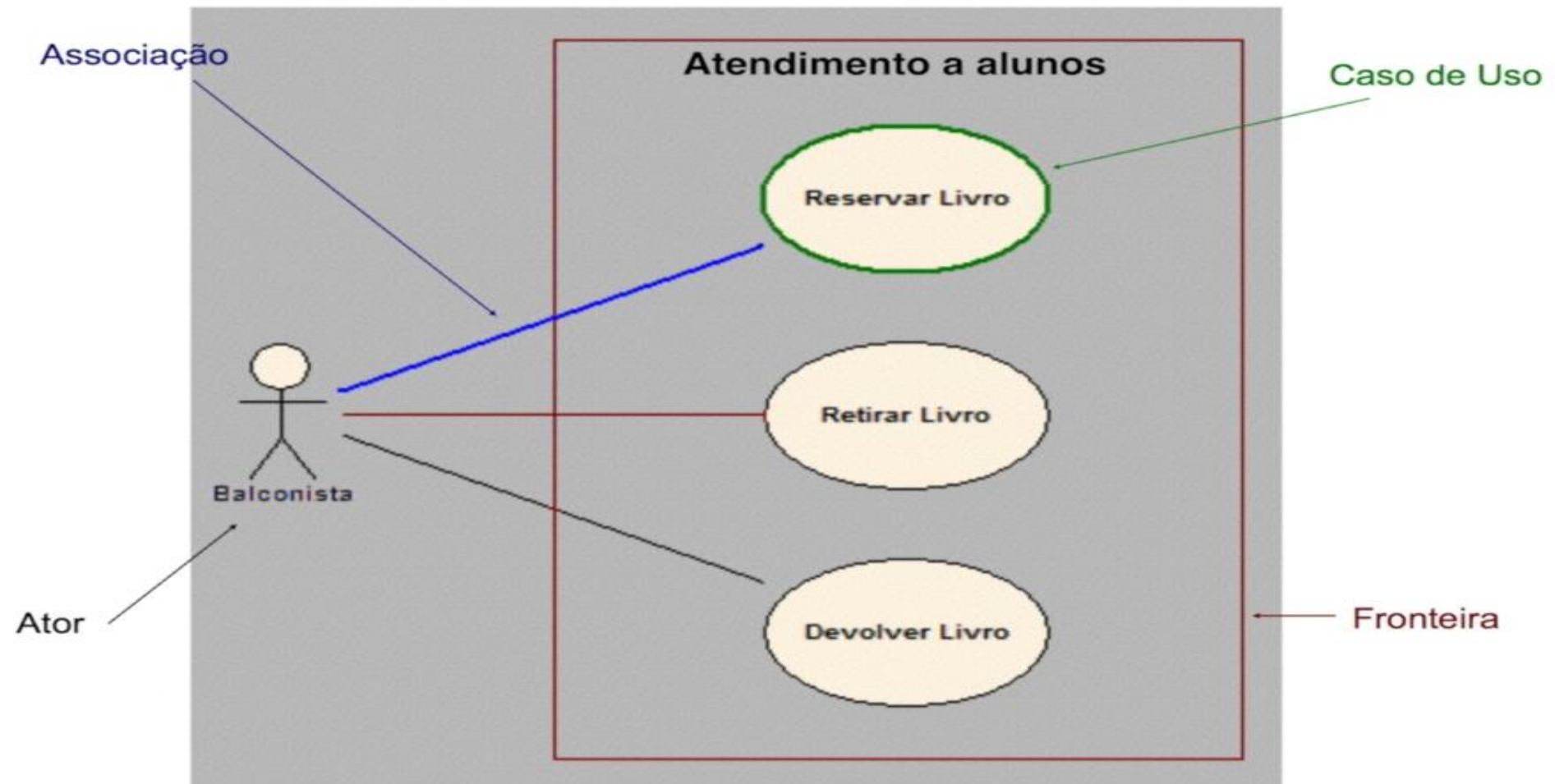


Diagrama de casos de USO

Relacionamentos

Inclusão (Include): (tem que fazer..)

- Se um caso de uso inicia ou inclui o comportamento de outro, dizemos que ele “inclui” outro caso de uso.
- Mostra que existe dependência entre os casos de uso;



Diagrama de casos de USO Inclusão

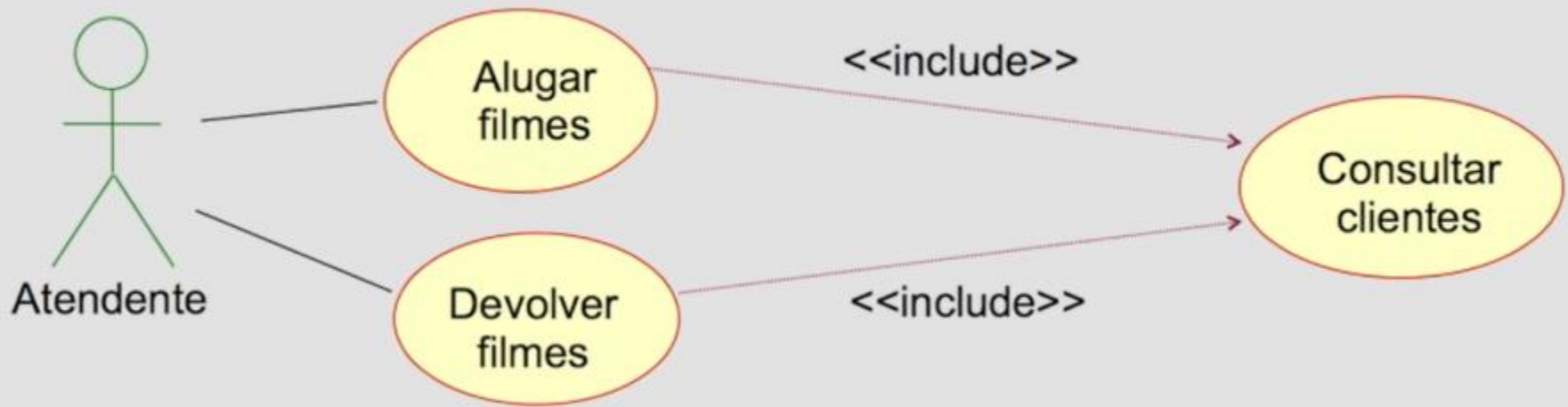


Diagrama de casos de USO

Relacionamentos

Extensão (Extends): (Pode fazer..)

- Adiciona comportamento a um caso de uso base;
- Os dois casos de uso são independentes;
- O caso de uso base pode ser executado mesmo sem a extensão.



Diagrama de casos de USO Extend

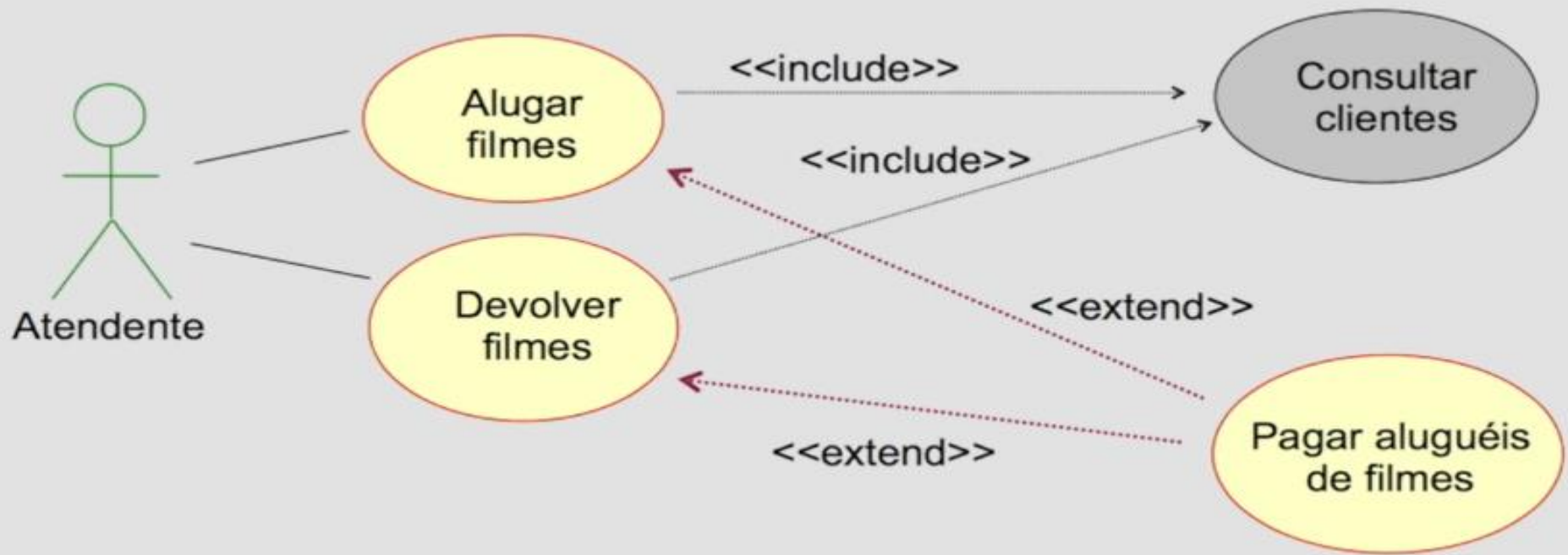


Diagrama de casos de USO *Generalização / especialização*

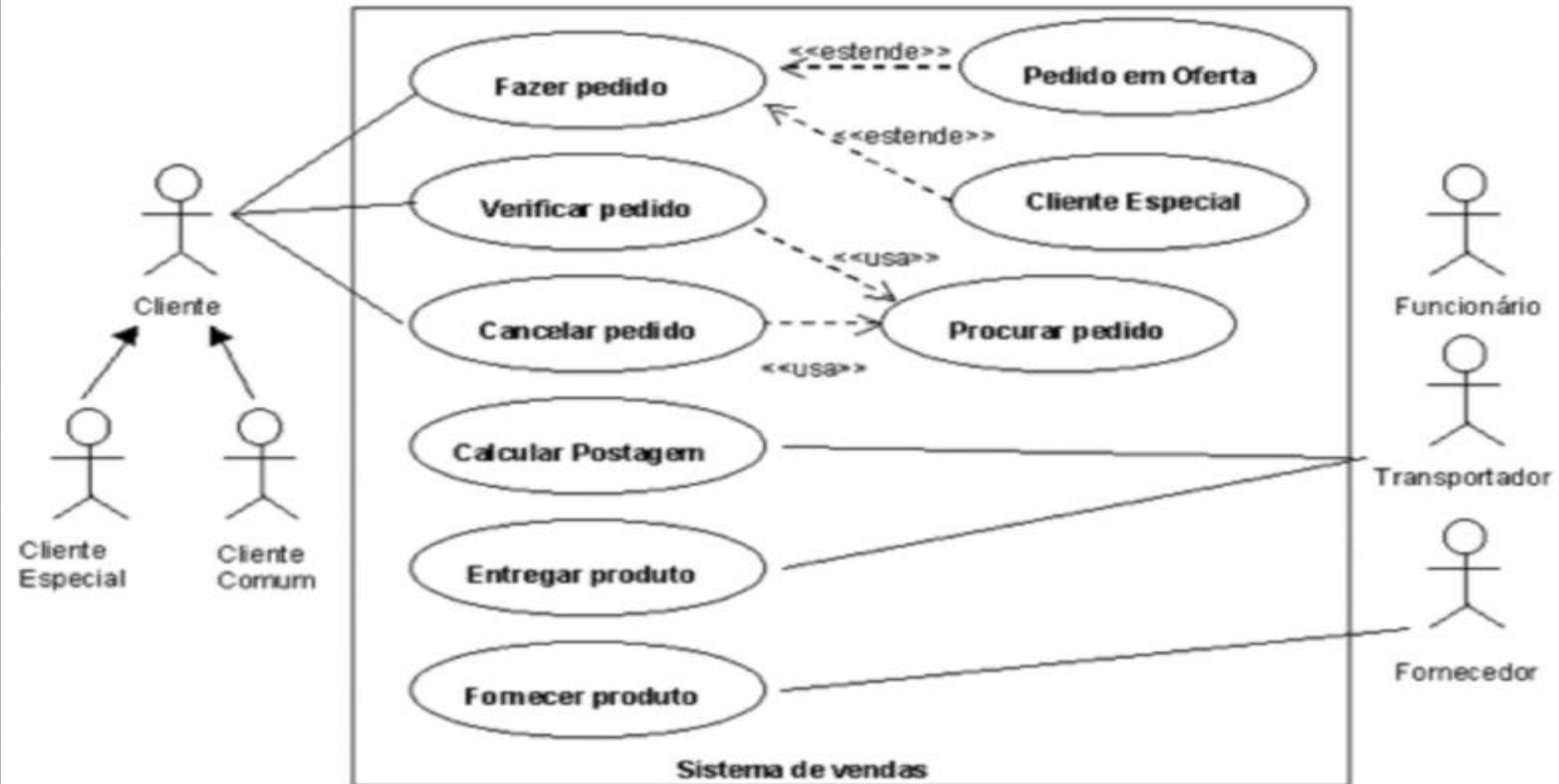
Um **caso de uso** base possui **diferentes especializações** e **generalizações** do seu comportamento.

A herança pode ser aplicada a casos de uso (funcionalidades) ou a atores (papéis).



Diagrama de casos de USO

exemplo: herança



Casos de Uso

Antes de mais nada, não confunda casos de uso, documentos textuais, com os diagramas de caso de uso. Apesar de tentarem falar sobre o mesmo tema é inquestionável que os documentos textuais de casos de uso são mais frequentemente usados em modelagem e documentação nas fases iniciais como na análise de requisitos.



Casos de Uso

Documento Descritivo

Documento mais detalhado de especificação de requisitos

- Uso não é tão comum com métodos ágeis
- Um ator realizando alguma operação com o sistema
- Incluem fluxo normal (feliz) e extensões
- Extensões:
 - Exceções (ou erros)
 - Detalhamento



Casos de Uso Documento Descritivo - Exemplo

Transferir Valores entre Contas

Ator: Cliente do Banco

Fluxo normal:

- 1 - Autenticar Cliente
- 2 - Cliente informa agência e conta de destino da transferência
- 3 - Cliente informa valor que deseja transferir
- 4 - Cliente informa a data em que pretende realizar a operação
- 5 - Sistema efetua transferência
- 6 - Sistema pergunta se o cliente deseja realizar uma nova transferência


Extensões:

- 2a - Se conta e agência incorretas, solicitar nova conta e agência
- 3a - Se valor acima do saldo atual, solicitar novo valor
- 4a - Data informada deve ser a data atual ou no máximo um ano a frente
- 5a - Se data informada é a data atual, transferir imediatamente
- 5b - Se data informada é uma data futura, agendar transferência

Fluxo feliz



Exceções e detalhamento

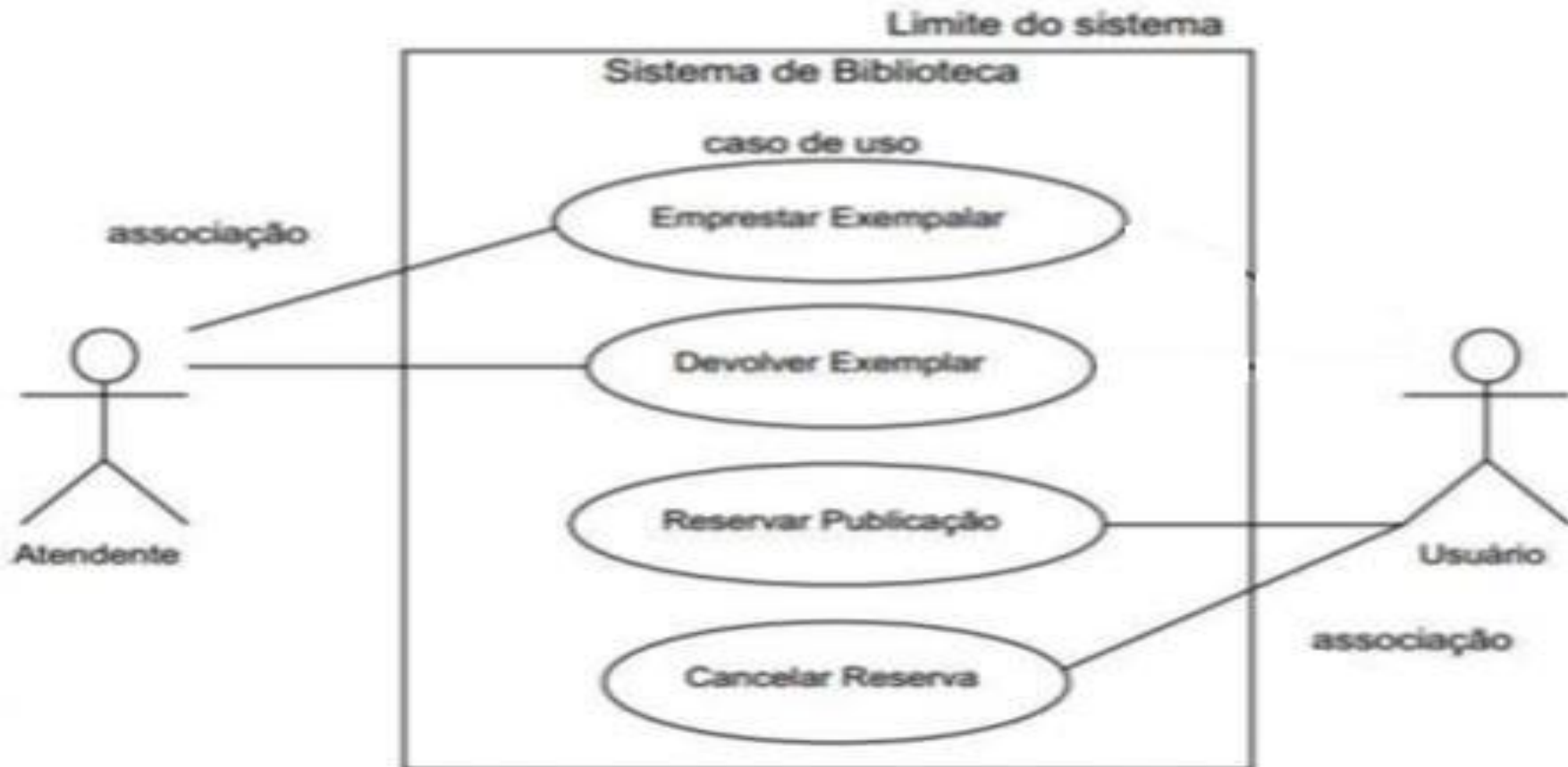


Casos de Uso Documento Descritivo - Modelo

UC001 – Nome do caso de uso	
Objetivo:	[Qual propósito?]
Requisitos:	[Quais requisitos funcionais?]
Atores:	[Quais atores?]
Prioridade:	[Qual prioridade de desenvolvimento?]
Pré-condições:	[Qual estado anterior do sistema?]
Frequência de uso:	[Qual frequência do uso?]
Pós-condições:	[Qual estado posterior do sistema?]
Campos:	[Quais campos serão necessários?]
Fluxo principal:	a) Ação 1. (A1) b) Ação 2. (A2)(E1) c) Caso de uso é encerrado.
Fluxo alternativo:	A1 – fluxo alternativo qualquer a) Ação 1. b) Ação 2. c) Volta ao passo “b” do fluxo principal.
	A2 – outro fluxo ...
Fluxo de exceção:	E1 – uma exceção a) Ação 1. b) Volta ao passo “a” do fluxo principal.
Validações:	[Como validar para saber se há exceção?]
Protótipo das telas:	

Exemplo: Biblioteca

UC: Emprestar Exemplar



Exemplo: Biblioteca Usando o Modelo

Código + Número sequenciado = ID

- UC001 UC001 - Emprestar exemplar

Nome do caso de uso (mesmo do diagrama)

- Emprestar exemplar

Indicar todos os envolvidos no processo

Separar com vírgula

Atendente, Usuário

Prioridade

Não confundir com frequência de uso

Cria uma ordem para ser programado

UC001 – Emprestar exemplar	P = 3 / Alta
UC002 – Devolver exemplar	P = 2 / Média
UC003 – Reservar publicação	P = 1 / Baixa
UC004 – Cancelar reserva	P = 1 / Baixa

Pré-Condições

São requisitos/estados que o sistema deve estar para que o caso aconteça

Pré-condição de “UC004 – Cancelar reserva” será a existência de uma reserva realizada em “UC003 – Reservar publicação”

Pós-condições

São requisitos/estados que o sistema deve estar após o caso acontecer

Aqui não tem “se”, é OBRIGATÓRIO

Pós-condição de “UC004 – Cancelar reserva” será a inexistência da reserva antes realizada em “UC003 – Reservar publicação”

Campos

Todas as características de todos os objetos não-atores envolvidos no caso

Objetos de “UC001 – Emprestar exemplar”

- Atendente (ator)
- Usuário (ator)
- Livro
 - Nome
 - Autor
 - ISBN
 - Quantidade de páginas
 - ...

Exemplo: Biblioteca usando o Modelo

FLUXO PRINCIPAL

Fluxos

Mecânica para todos os fluxos

- a) Ator faz alguma coisa
- b) Sistema responde
- c) Ator faz outra
- d) Sistema responde
- e) O caso de uso é encerrado

a)Usuário: Informa o nome do livro que deseja emprestar ao Atendente.

b)Atendente: Digita o nome do livro no sistema.

c)Sistema: Busca o livro no banco de dados e retorna informações sobre a disponibilidade e retorna mensagem ao atendente

d)Atendente: Informa ao Usuário a disponibilidade do livro.

.....

.....



Exemplo: Biblioteca usando o Modelo

Fluxo alternativo

Pode apontar para outro fluxo alternativo e de exceção

Pode encerrar em si mesmo

Pode voltar para o fluxo principal

FLUXO ALTERNATIVO

Livro não disponível

a) Sistema: Exibe uma mensagem de indisponibilidade para o Atendente.

b) Atendente: Informa ao Usuário que o livro não está disponível, e pode sugerir alternativas (ex: outros livros do mesmo autor, livros sobre o mesmo tema).

c) Usuário: Escolhe uma alternativa ou desiste do empréstimo.

....

n) Sistema: retorna ao passo “b” do fluxo principal



Exemplo: Biblioteca usando o Modelo

Fluxo de exceção

Pode apontar para outro fluxo alternativo e de exceção

Pode encerrar em si mesmo

Pode voltar para o fluxo principal

FLUXO EXCEÇÃO

Livro não retorna se disponível ou não:

a) Sistema: não retorna nenhuma informação sobre a disponibilidade ou não do livro

b) Atendente: verifica a conexão com o banco de dados

....

n) Atendente: encerra o sistema



Exemplo: Biblioteca usando o Modelo

VALIDAÇÃO

Para nossa validação utilizaríamos algoritmos que disparassem e testassem os fluxos de exceção do nosso sistema

PROTÓTIPO

Crie designers de telas que ajudarão a entender melhor as interfaces/telas requeridas pelo usuário.



Exercício 1

Usando o modelo apresentado crie os casos de uso descritivos do restante dos requisitos funcionais da biblioteca.



Exercício 2

Modele através do diagrama de caso de uso o sistema de cadastro de alunos de uma universidade

Atores:

- Aluno: Realiza o cadastro, consulta suas informações, altera seus dados e efetua a matrícula em cursos.
- Secretário: Cadastra alunos, consulta informações dos alunos e efetua a matrícula dos alunos em cursos.



Exercício 2

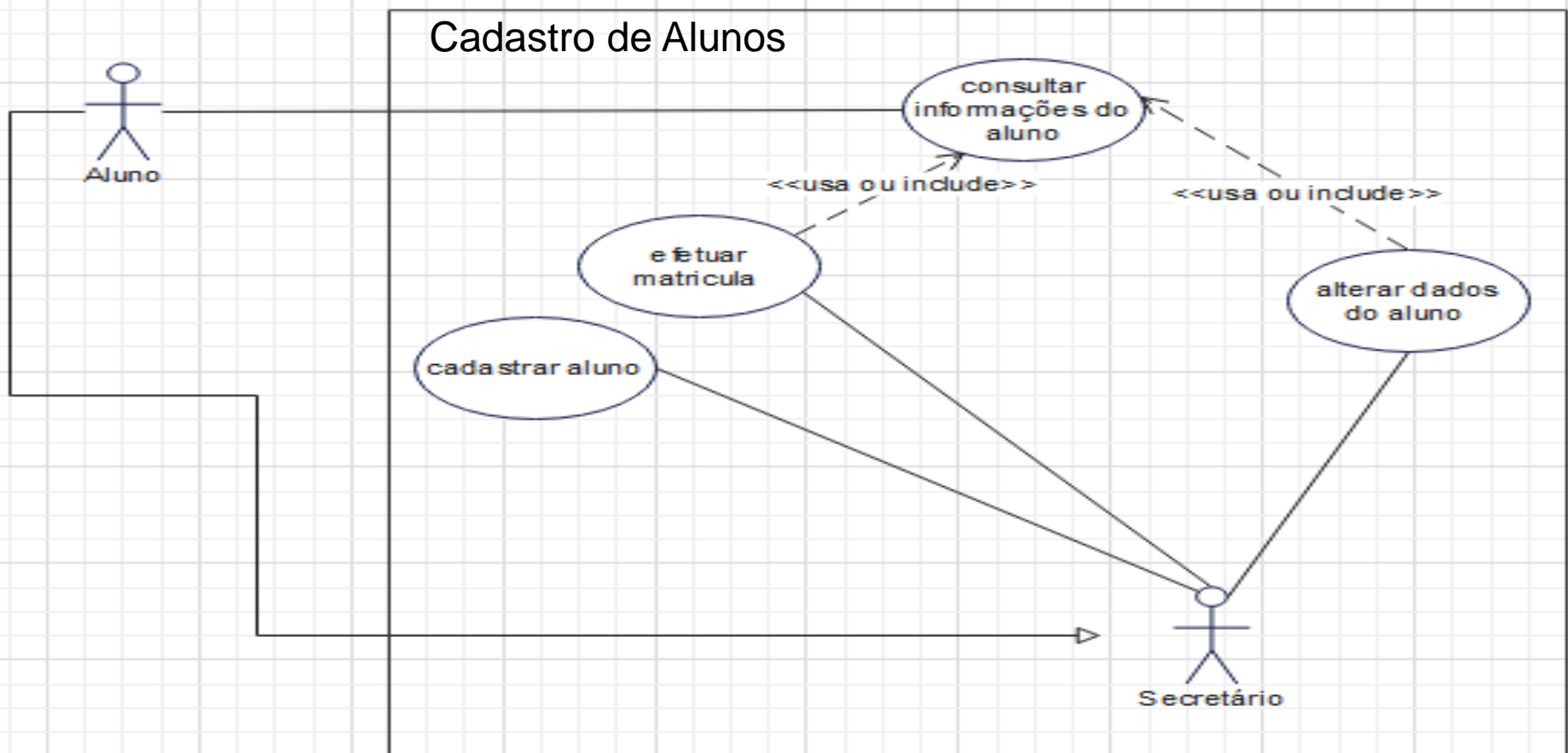
Casos de Uso:

- Cadastrar Aluno: O aluno informa seus dados pessoais e o sistema os registra.
- Consultar Informações do Aluno: O aluno ou o secretário consulta as informações cadastrais do aluno.
- Alterar Dados do Aluno: O aluno ou o secretário altera seus dados pessoais.
- Efetuar Matrícula: O aluno ou o secretário efetua a matrícula do aluno em um curso. (dica: <https://app.diagrams.net/>, ou Lucidchart)



Exercício 2

Proposta resolução



Exercício 3

Entrega em duplas no forms

Uma loja possui discos para venda. Um cliente pode comprar uma quantidade ilimitada de discos para isto ele deve se dirigir à loja. A loja possui um atendente cuja função é atender os clientes durante a venda dos discos. A loja também possui um gerente cuja função é administrar estoque para que não falem discos. Além disso é ele quem dá folga ao atendente, ou seja, ele também atende os clientes durante a venda dos discos.

As vendas podem ser à vista ou a prazo. Em ambos os casos o estoque é atualizado e uma nota fiscal, entregue ao consumidor.



Exercício 3

Entrega em duplas no forms

No caso de uma venda à vista, clientes cadastrados na loja ganham um desconto de 5%.

No caso de uma venda a prazo, ela pode ser parcelada em 4 pagamentos com um acréscimo de 10%. As vendas a prazo podem ser pagas no cartão ou no boleto. Para pagamento com boleto, são gerados boletos bancários que são entregues ao cliente e armazenados no sistema para lançamento posterior no caixa. Para pagamento com cartão, na compra de mais de 5 discos será concedido o mesmo desconto das compras à vista.



Exercício 3

Entrega em duplas no forms

Dicas:

- Identifique os atores (só aqueles que interagem com o sistema)
- Identifique os casos de uso (comece sem as vendas a prazo, com cartão, etc..)
- Relacionamentos:
 - associação
 - herança
 - inclusão ou dependência
- Fronteira do sistema

Elaborar o diagrama de casos de uso e também o casos de uso no documento descritivo conforme o modelo apresentado nessa aula. *Para os atores além da função colocar também os nomes dos participantes do grupo.* Salvar em PDF e enviar para o forms que será aberto. *Em caso de plágio serão zerados todas as atividades iguais.* Exemplo

