Programação Avançada

Revisão
Modelagem orientada a objetos

Prof. Chauã Queirolo

Sumário

- Orientação a objetos
  - Objetos
  - Classes
  - Encapsulamento
  - Herança
  - Polimorfismo
- UML

Paradigmas de programação

Paradigmas de programação

Introdução

- **Visão** do **programador** em relação aos **programas**
  - Estrutura
  - Execução
- Principais **paradigmas**:
  - **Imperativo:** estruturado, procedural, orientado a objetos
  - **Declarativo:** funcionalista, lógico

Paradigmas de programação

Paradigma orientado a objetos

- **Descreve** o sistema com elementos do **mundo real**
- **Considera** que todas as **componentes** são **objetos**
- **Objetos** possuem sua **estrutura** e desempenham **ações**
- **Classificados** de acordo com suas **características**
- **Exemplo**: Java, C++, C#, Python

Paradigmas de programação

Paradigma orientado a objetos

- **Vantagens**
  - Abstração
  - Modularização
  - Extensibilidade
  - Reaproveitamento de código

Orientação a objetos

Objetos

Conceitos

- O **universo** é formado por **objetos**

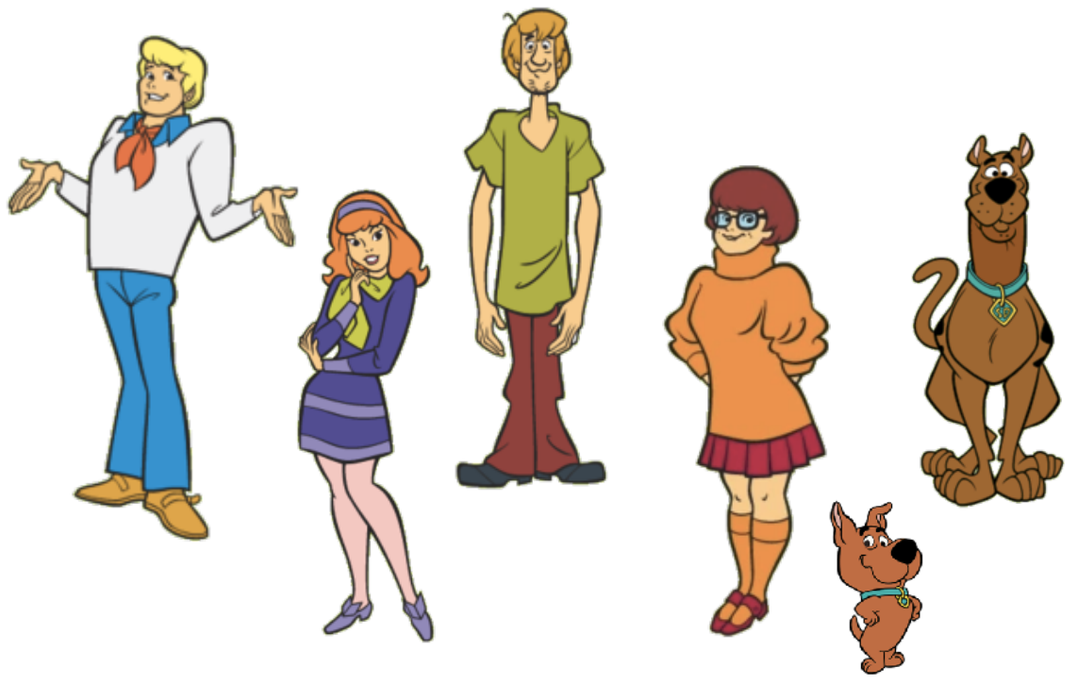
- Cada **objeto** possui:
  - **Características**
  - **Funções**

Classes

Conceitos

- Os **objetos** são **classificados** de acordo com
  - **Características** semelhantes
  - **Funcionalidades** semelhantes

- **Classes** são **agrupamentos** objetos **semelhantes** entre si

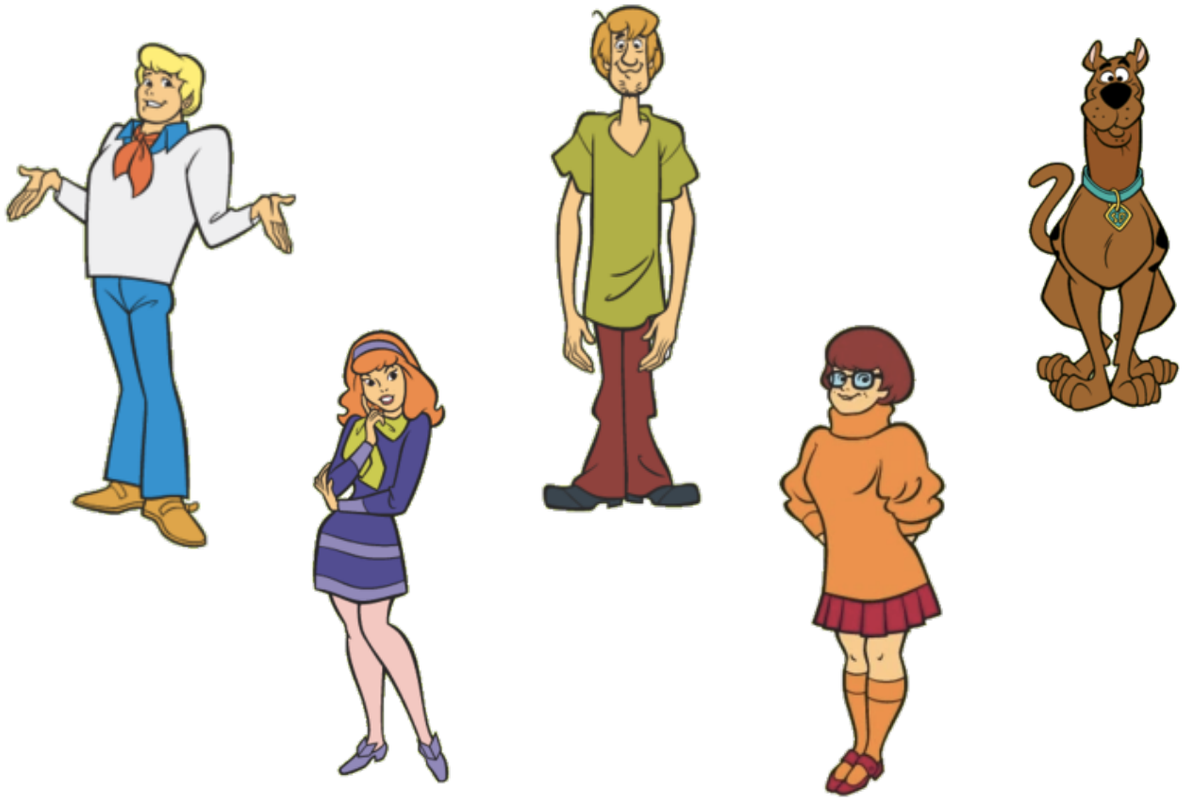


Classes

Conceitos

- **Atributos:** características, propriedades
- **Métodos:** funcionalidades, ações, procedimentos

Pessoa
- nome - idade - peso - altura
+ andar() + conversar() + dormir() + dirigirCarro()



Classes

Relacionamentos

- Associação
- Agregação
- Composição
- Herança

Pilares da Orientação a Objetos

- Encapsulamento
- Herança
- Polimorfismo

Encapsulamento

- Todo objeto é responsável pelos seus atributos
- Esconder do mundo externo:
  - Estrutura interna dos objetos
  - Detalhes de implementação
- Interação através de uma interface pública

Herança

- Estabelece a relação **é um** entre duas classes
- Permite
  - Abstração
  - Reaproveitamento de código

Polimorfismo

- Um objeto pode assumir **diferentes formas**
- **Tipos** de polimorfismo:
  - **por inclusão** - via herança
  - **paramétrico** - tipos genéricos
  - **sobrescrita** - redefinição de métodos
  - **sobrecarga** - métodos com mesmo nome, parâmetros diferentes

UML

UML

- A **UML** é uma linguagem de **modelagem visual**
- **Objetivos**:
  - Modelagem de processos de negócios
  - Análise, projeto e implementação de sistemas de software

NÃO define processo para desenvolvimento de software!

Diagrama de casos de uso

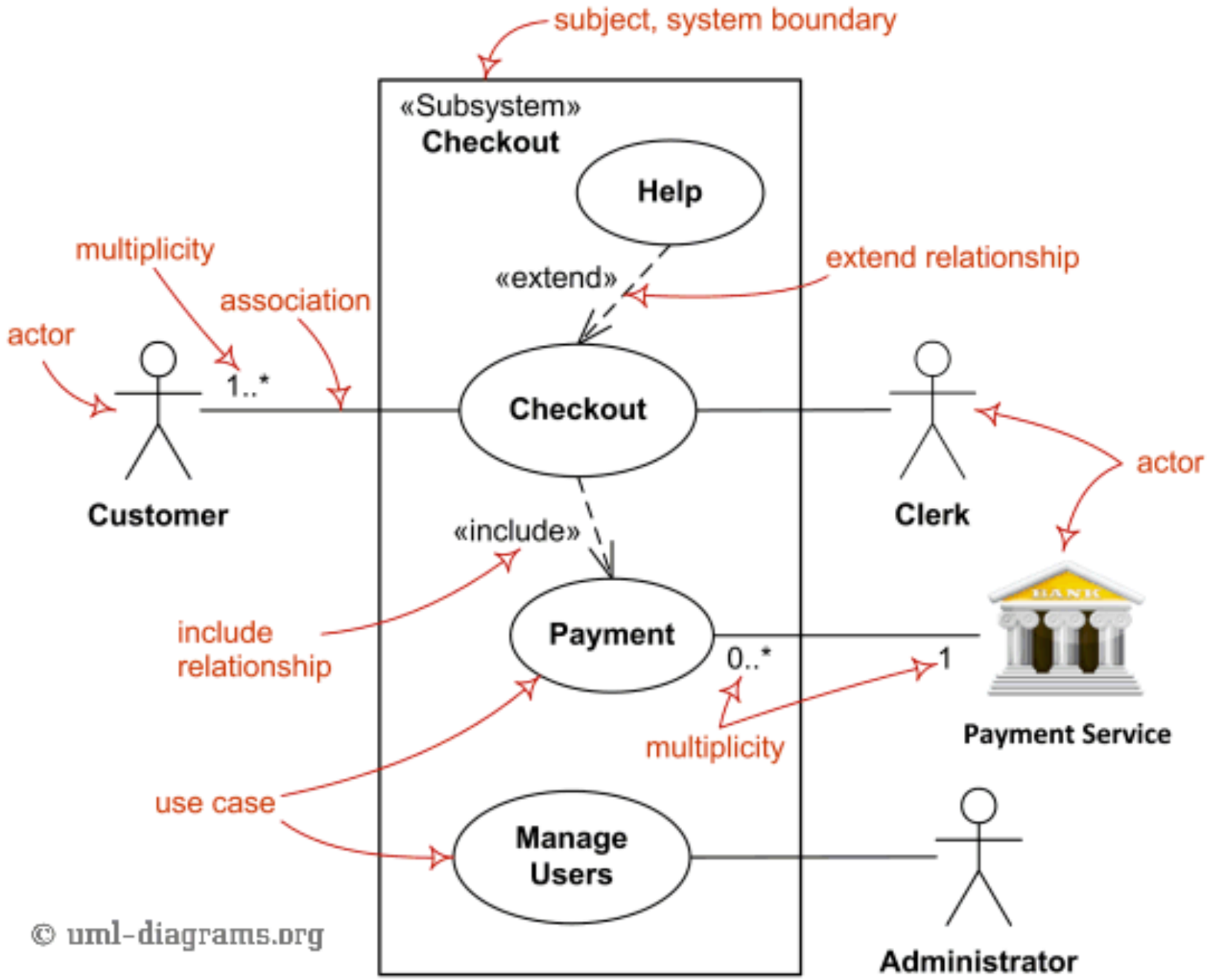


Diagrama de casos de uso

Caso de uso:
Finalizar Compra

Referências:
RF_F2

Descrição Geral:
O caso de uso inicia-se quando cliente deseja efetuar compra dos produtos que estão inseridos no carrinho de compras.

Atores:
Cliente.

Pré-condições:
Cliente logado no sistema, produtos já inseridos no carrinho de compras.

Garantia de sucesso (Pós-condições):
Pedido fechado, compra efetuada, sistema aguardando confirmação de pagamento.

Requisitos Especiais:

Fluxo Básico:

1. Cliente deseja finalizar compra, sistema solicita que informe a forma de pagamento e de entrega.
2. Cliente deseja efetuar pagamento em forma de cartão crédito/débito.
3. Sistema solicita informações do cartão do cliente.
4. Sistema faz validação das informações.
5. Sistema gera o número do pedido.
6. Compra finalizada com sucesso.

Fluxo Alternativo:

1. Cliente deseja efetuar pagamento através de boleto bancário.
    1. Sistema gera o boleto para o cliente. Retorna ao passo 5.

Diagrama de classes

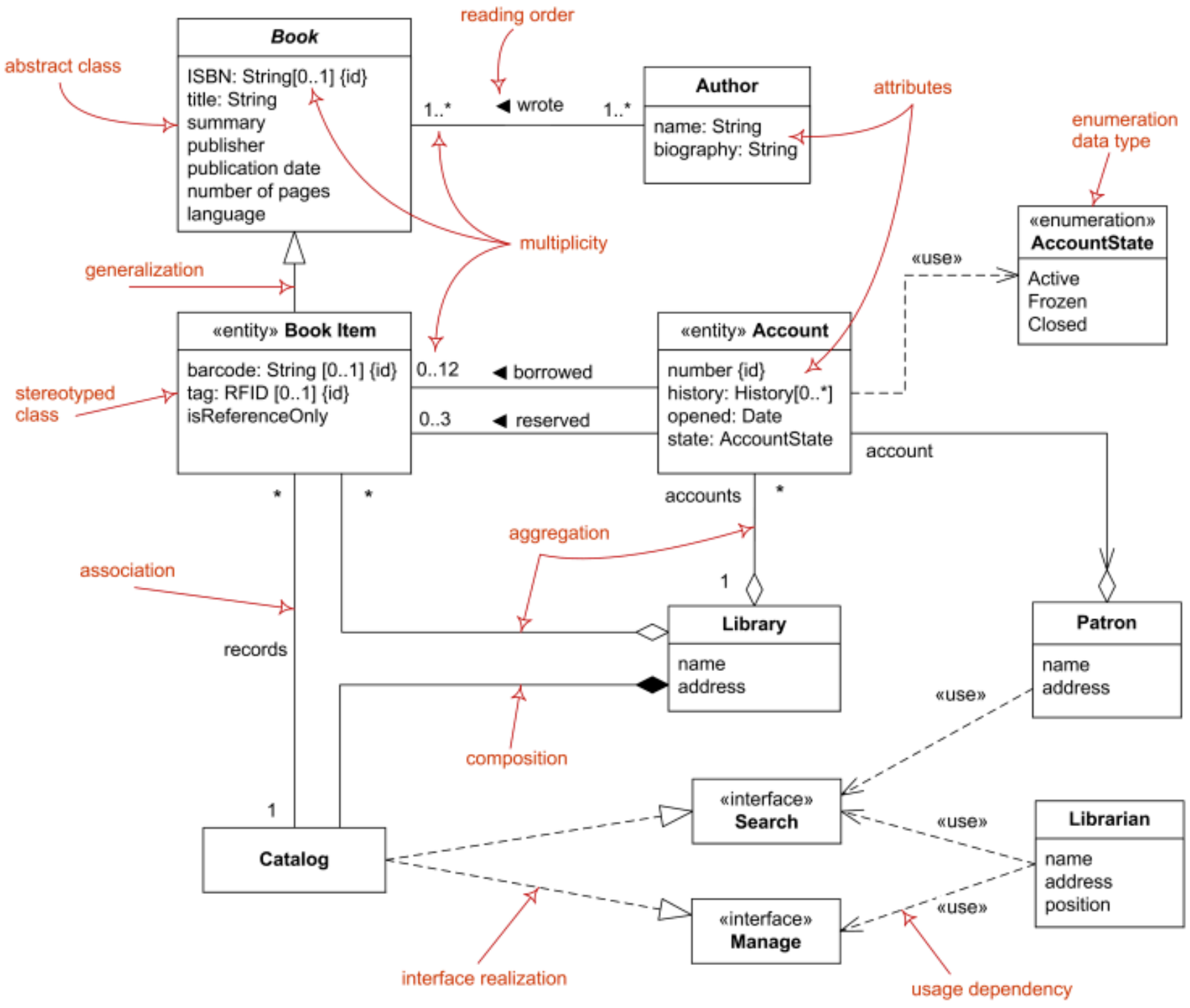


Diagrama de sequências

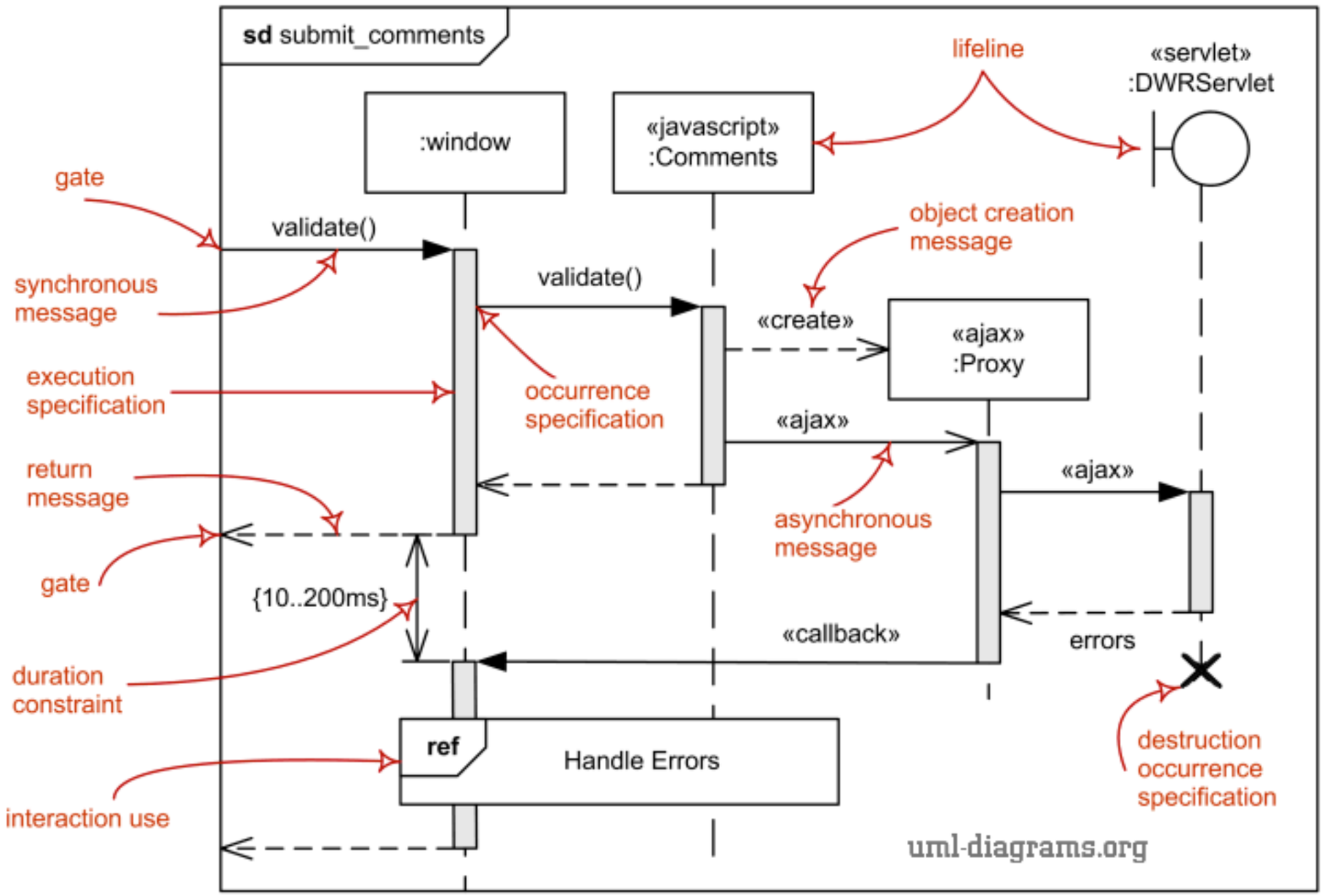
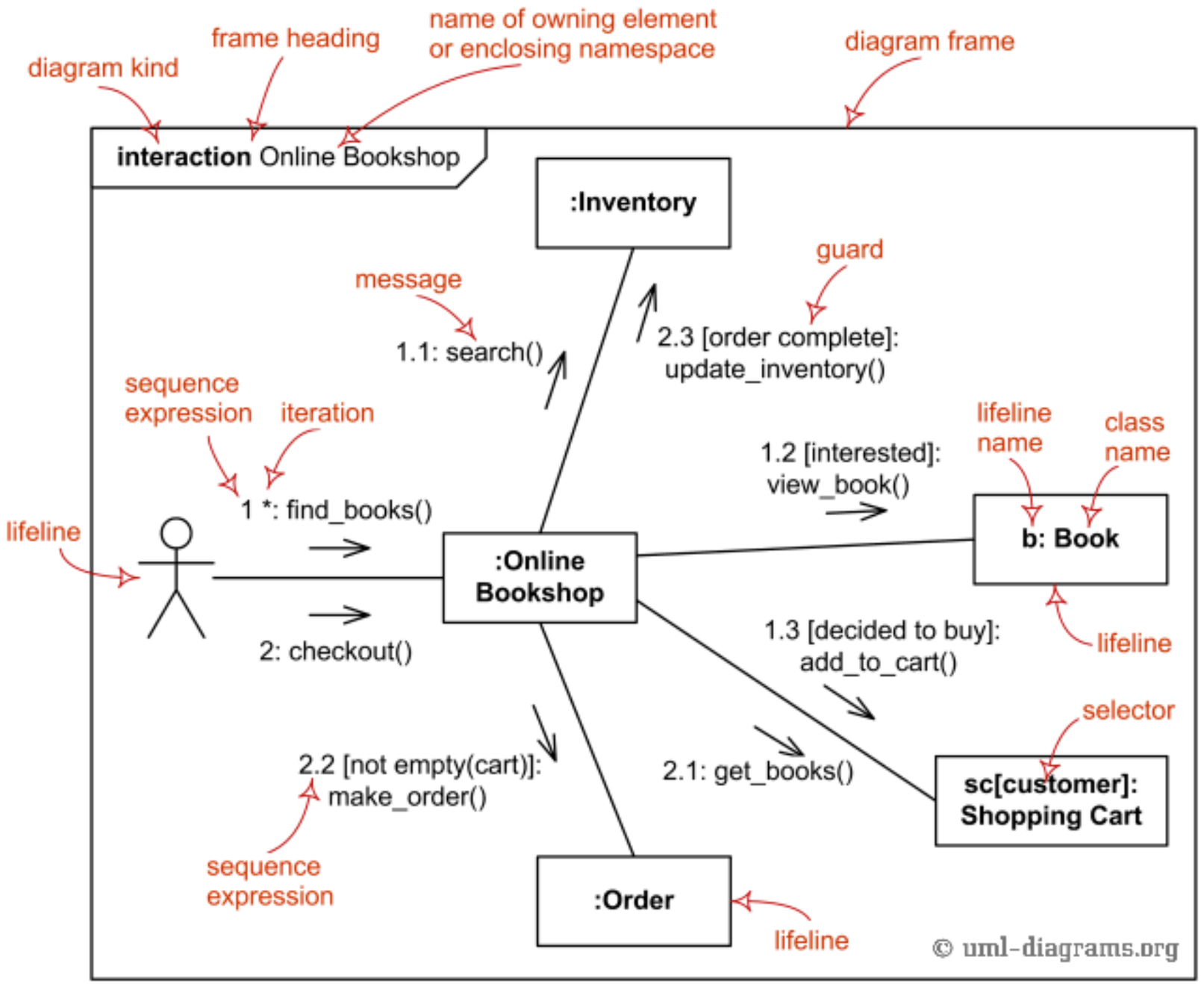


Diagrama de comunicação



Referências bibliográficas

📚 Referência Bibliográficas

- **Especificação da UML:** https://www.omg.org/spec/UML/
- **Diagramas da UML:** https://www.uml-diagrams.org/