



Instituto Tecnológico de Aeronáutica

CES-22 - Programação Orientada a Objetos

Alunos: Rafael Studart Mattos Di Piero

Matéria: CES-22

Professor(a): Edgar Yano

Laboratório 2 - Documentação

1.) Modelagem do Problema

Inicialmente, pode-se analisar as especificações do problema, apresentadas na **Figura 1** abaixo:

Uma livraria vende diferentes produtos, mas o principal produto é o livro.

Livros possuem os seguintes atributos: título, autor, gênero, edição, editora, preço de venda, preço de compra, e impostos que dependem do tipo de gênero e a diferença entre preço de venda e de compra.

Autores possuem um cadastro com nome, e-mail para contato e títulos publicados.

Cada Cliente da livraria também é cadastrado com nome, e-mail e compras passadas.

Cada compra de cliente gera uma lista de produtos adquiridos com quantidade e respectivos valores de cada item adquirido.

Figura 1: Especificações do Problema

Analisando as especificações e considerando os princípios básicos de OO e SOLID, elaborou-se uma modelagem, apresentada por meio do diagrama de classes exposto na **Figura 2**.

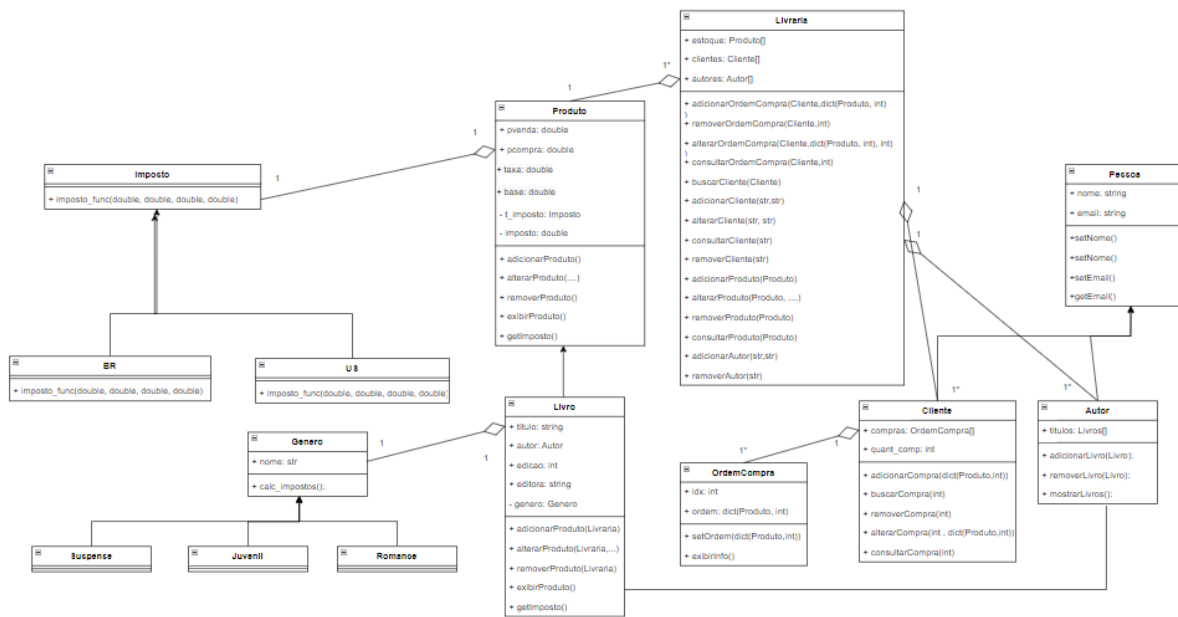


Figura 2: Diagrama de Classes da Modelagem

Em seguida, algumas decisões de projeto serão explicadas:

- Decidiu-se implementar uma classe interface **Produto**, de maneira que a classe **Livraria** contenha produtos e não somente livros, considerando que pode haver expansão nos negócios de um livraria, vendendo produtos diversos.
- Decidiu-se implementar uma classe interface **Pessoa**, da qual as classes **Cliente** e **Autor** herdam, visto que apresentam atributos iguais.
- Tendo em vista que toda ordem de compra (classe **OrdemCompra**) está associada com um cliente (classe **Cliente**), decidiu-se fazer com que a classe contenha uma lista de ordens de compra.
- As operações de adição, remoção, alteração e consulta de livros foram implementadas considerando que qualquer produto deve possuir métodos análogos, descritos na interface **Produto**, com cada produto implementando suas variações específicas. Para o método de alteração de livros, considerou-se que atualmente ele muda somente o título do livro, por simplicidade, visto que outras mudanças poderiam ser implementadas sem erro na execução.
- As operações de remoção, alteração e consulta de clientes são realizadas pelo nome do cliente em questão, visto que é a maneira mais intuitiva e comum dessas operações serem realizadas. A operação de adição recebe um nome e um email para ser gerado um novo objeto da classe **Cliente**.
- As operações de adição, remoção, alteração e consulta de ordens foram implementadas considerando um determinado cliente, visto que uma ordem de compra sempre está vinculada com uma cliente. Deve-se destacar que associou-se um índice a cada ordem de compra, visto que essa seria a maneira coerente de organizar os dados, realizando a busca em função desse índice.
- Decidiu-se implementar a classe **Gênero**, tal que os livros possuem um atributo gênero com esse tipo. Isso foi escolhido para permitir uma adição de qualquer gênero e as variações de regras que são associadas ao cálculo de impostos.

- Criou-se uma classe **Imposto** que determina as regras de cálculo de impostos, podendo ser adicionadas e alteradas com base em localidade ou ano, ou qualquer outra mudança na forma de cálculo de impostos.
- Considerou-se que o cálculo de impostos depende de quatro variáveis, o preço de venda, o preço de compra, uma taxa e uma base. Como esses valores se relacionam, estão definidos por meio da classe **Imposto**, enquanto a taxa e base são definidos em cada produto. Para **Livros**, decidiu-se implementar taxas e bases diferentes com base em cada gênero, visto que foi pedido para cada gênero alterar o cálculo de impostos.

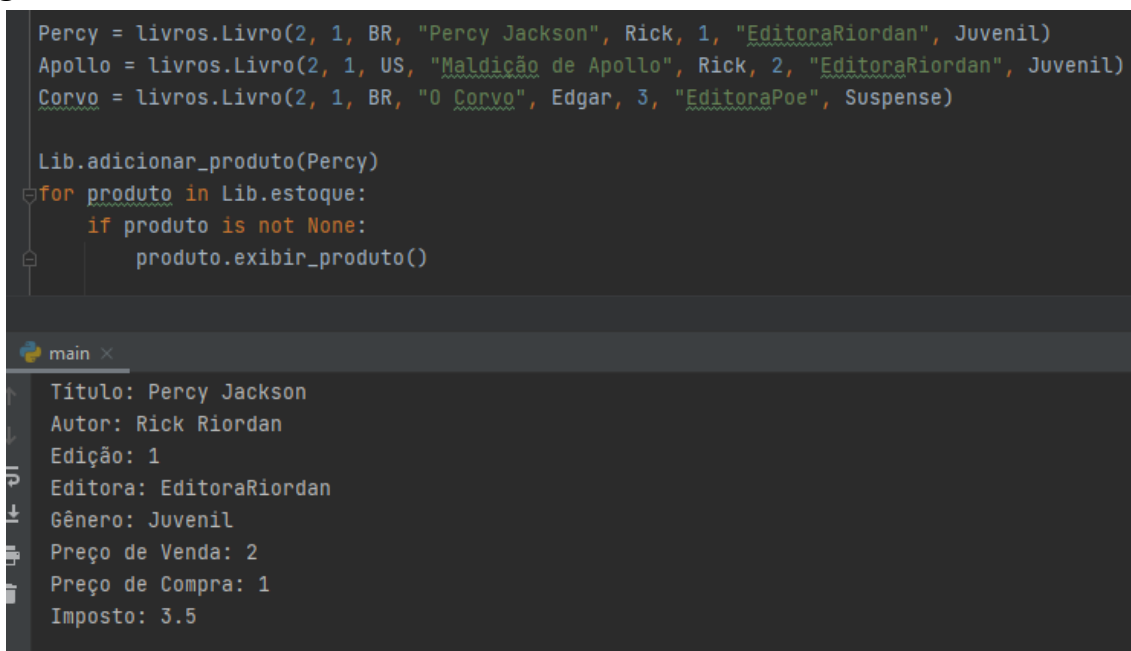
2.) Implementação do Problema

Considerando a modelagem apresentada anteriormente, implementou-se um protótipo do funcionamento da livraria, utilizando a linguagem *Python*.

Para testar o funcionamento, deve-se destacar o código implementado no arquivo *main.py*, o qual mostra alguns casos de testes para cada item pedido.

Item A)

Para testarmos as operações referentes às operações com livros, pode-se observar as **Figuras 3 à 6** abaixo:



```
Percy = livros.Livro(2, 1, BR, "Percy Jackson", Rick, 1, "EditoraRiordan", Juvenil)
Apollo = livros.Livro(2, 1, US, "Maldição de Apollo", Rick, 2, "EditoraRiordan", Juvenil)
Corvo = livros.Livro(2, 1, BR, "O Corvo", Edgar, 3, "EditoraPoe", Suspense)

Lib.adicionar_produto(Percy)
for produto in Lib.estoque:
    if produto is not None:
        produto.exibir_produto()
```

main ×

```
Título: Percy Jackson
Autor: Rick Riordan
Edição: 1
Editora: EditoraRiordan
Gênero: Juvenil
Preço de Venda: 2
Preço de Compra: 1
Imposto: 3.5
```

Figura 3: Teste do Método de Adição de Livros

Pode-se perceber que o método de adição de livros funciona corretamente, tendo em vista que ao analisar o estoque da livraria, conseguimos obter o produto e seus dados corretamente.

```
Lib.alterar_produto(Percy, "Percy Jackson e os Olimpianos")

for produto in Lib.estoque:
    produto.exibir_produto()
```

main ×

Título: Percy Jackson e os Olimpianos
Autor: Rick Riordan
Edição: 1
Editora: EditoraRiordan
Gênero: Juvenil
Preço de Venda: 2
Preço de Compra: 1
Imposto: 3.5

Figura 4: Teste do Método de Alteração de Livros

Pode-se perceber que o método de alteração de livros proposto funciona corretamente, tendo em vista que ao analisar o estoque da livraria, conseguimos obter o produto e seus dados, considerando a alteração no título pedida.

```
for produto in Lib.estoque:
    produto.exibir_produto()

Lib.consultar_produto(Percy)
Lib.consultar_produto(Apollo)
#
```

main ×

Título: Percy Jackson e os Olimpianos
Autor: Rick Riordan
Edição: 1
Editora: EditoraRiordan
Gênero: Juvenil
Preço de Venda: 2
Preço de Compra: 1
Imposto: 3.5

Produto não encontrado

Figura 5: Teste do Método de Consulta de Livros

Pode-se perceber que o método de consulta de livros funciona corretamente, tendo em vista que ao analisar ao requisitar a consulta de um livro que temos no estoque da loja, obtemos as suas informações, enquanto o resultado para um livro não contido em estoque é “Produto não encontrado”, como esperado.

```
Lib.adicionar_produto(Apollo)

Lib.consultar_produto(Apollo)

Lib.remover_produto(Apollo)

Lib.consultar_produto(Apollo)
```

```
main x
↑ Título: Maldição de Apollo
↓ Autor: Rick Riordan
: Edição: 2
: Editora: EditoraRiordan
: Gênero: Juvenil
: Preço de Venda: 2
: Preço de Compra: 1
: Imposto: 1.5

Produto não encontrado
```

Figura 6: Teste do Método de Remoção de Livros

Pode-se perceber que o método de remoção de livros funciona corretamente, dado que ao removermos um produto que está contido no estoque, confirmado pelo resultado da consulta, recebemos a confirmação que ele foi removido do estoque, por meio de uma segunda consulta.

Item B)

Para testar as operações referentes aos clientes, pode-se observar as **Figuras 7 à 10** abaixo:

```
Lib.adicionar_cliente("Rafael", "rafael@hotmail.com")

for cliente in Lib.clientes:
    print(cliente.get_nome(), cliente.get_email())
```

```
main x

Rafael rafael@hotmail.com
```

Figura 7: Teste do Método de Adição de Clientes

Ao analisar o resultado, pode-se perceber que o método está com o comportamento correto, pois ,após a sua adição, o cliente e seus dados puderam ser acessados por meio do "cadastro" dos clientes da livraria.

```
Lib.adicionar_cliente("Rafael", "rafael@hotmail.com")

for cliente in Lib.clientes:
    print(cliente.get_nome(), cliente.get_email())
print("\n")
Lib.alterar_cliente("Rafael", "rafael@gmail.com")

for cliente in Lib.clientes:
    print(cliente.get_nome(), cliente.get_email())

for produto in Lib.estoque
```



```
main x
Imposto: 0.0

Rafael rafael@hotmail.com

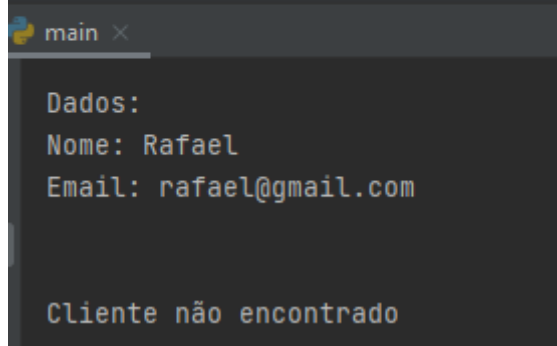
Alterado com sucesso.

Rafael rafael@gmail.com
```

Figura 8: Teste do Método de Alteração de Clientes

Ao analisar o resultado, pode-se perceber que o método está com o comportamento correto, pois foi realizada uma busca por meio do nome do cliente, encontrando-o e alterando o seu campo de email, exatamente com o esperado.

```
Lib.consultar_cliente("Rafael")
print("\n")
Lib.consultar_cliente("Bruno")
```



```
main x

Dados:
Nome: Rafael
Email: rafael@gmail.com

Cliente não encontrado
```

Figura 9: Teste do Método de Consulta de Clientes

Ao analisar o resultado, pode-se perceber que o método está com o comportamento correto, pois foi realizada uma busca por meio do nome do cliente, mostrando suas

informações, caso exista no cadastro, e informando-o caso um cliente com esse nome não exista.



```
Lib.consultar_cliente("Rafael")
print("\n")
Lib.remover_cliente("Rafael")
Lib.consultar_cliente("Rafael")
```

main x

Dados:
Nome: Rafael
Email: rafael@gmail.com

Removido com sucesso
Cliente não encontrado

Figura 10: Teste do Método de Remoção de Clientes

Ao analisar o resultado, pode-se perceber que o método está com o comportamento correto, pois, para um cliente existente, a remoção conseguiu eliminar o cliente do cadastro, como observado pela consulta realizada em seguida da remoção.

Item C)

Por fim, para a análise dos métodos de ordens de compra, analisou-se os casos de teste expostos nas **Figuras 11 a 15**.

```
Lib.adicionar_ordemcompra("Rafael", ordemCompra)
for cliente in Lib.clientes:
    for compra in cliente.compras:
        compra.exibir_info()
print("\n")
```



```
main x
Qtd: 1
Título: Percy Jackson e os Olimpianos
Autor: Rick Riordan
Edição: 1
Editora: EditoraRiordan
Gênero: Juvenil
Preço de Venda: 2
Preço de Compra: 1
Imposto: 3.5

Qtd: 2
Título: Maldição de Apollo
Autor: Rick Riordan
Edição: 2
Editora: EditoraRiordan
Gênero: Juvenil
Preço de Venda: 2
Preço de Compra: 1
Imposto: 1.5
```

Figura 11: Teste do Método de Adição de Ordem de Compra

Ao observar a figura acima, pode-se perceber que o comportamento está como o esperado, visto que ,após a operação de adição, pode-se iterar pela lista de cliente, analisando a lista de ordem de compra de cada um deles, obtendo as informações necessárias.


```
Lib.alterar_ordemcompra("Rafael", ordemCompra2, 1)
for cliente in Lib.clientes:
    for compra in cliente.compras:
        compra.exibir_info()
print("\n")
```



```
main x
Qtd: 3
Título: Percy Jackson e os Olimpianos
Autor: Rick Riordan
Edição: 1
Editora: EditoraRiordan
Gênero: Juvenil
Preço de Venda: 2
Preço de Compra: 1
Imposto: 3.5

Qtd: 1
Título: Maldição de Apollo
Autor: Rick Riordan
Edição: 2
Editora: EditoraRiordan
Gênero: Juvenil
Preço de Venda: 2
Preço de Compra: 1
Imposto: 1.5
```

Figura 12: Teste do Método de Alteração de Ordem de Compra

Ao observar a figura acima, pode-se perceber que o comportamento está como o esperado, visto que, ao passar o nome do cliente, foi possível alterar a lista da ordem de compra que estava contida no índice 1.

```
Lib.alterar_ordemcompra("Rafael", ordemCompra2, 2)
for cliente in Lib.clientes:
    for compra in cliente.compras:
```



```
main x
Ordem de Compra com esse índice não encontrada
```

Figura 13: Teste do Método de Alteração de Ordem de Compra(Caso de Borda)

Deve-se destacar que, caso o índice não exista, é mostrada uma mensagem e nenhuma alteração é realizada.

```
Lib.consultar_ordemcompra("Rafael", 1)

for cliente in Lib.clientes:
    for compra in cliente.compras:
        if cliente.nome == "Rafael" and compra.index == 1:
            print(f'Qtde: {compra.quantidade}
            Título: {compra.titulo}
            Autor: {compra.autor}
            Edição: {compra.edicao}
            Editora: {compra.editora}
            Gênero: {compra.genero}
            Preço de Venda: {compra.preco_venda}
            Preço de Compra: {compra.preco_compra}
            Imposto: {compra.imposto}')

main x
```

```
Qtde: 3
Título: Percy Jackson e os Olimpianos
Autor: Rick Riordan
Edição: 1
Editora: EditoraRiordan
Gênero: Juvenil
Preço de Venda: 2
Preço de Compra: 1
Imposto: 3.5

Qtde: 1
Título: Maldição de Apollo
Autor: Rick Riordan
Edição: 2
Editora: EditoraRiordan
Gênero: Juvenil
Preço de Venda: 2
Preço de Compra: 1
Imposto: 1.5
```

Figura 14: Teste do Método de Consulta de Ordem de Compra

Ao observar a figura acima, pode-se perceber que o comportamento está como o esperado, visto que ao consultar um índice e o nome de um cliente, pode-se determinar as informações da ordem de compra. Destaca-se que o mesmo comportamento da **Figura 13** ocorre quando o índice passado é inválido.

```
Lib.consultar_ordemcompra("Rafael", 1)
Lib.remover_ordemcompra("Rafael", 1)
Lib.consultar_ordemcompra("Rafael", 1)

main x
Título: Percy Jackson e os Olimpianos
Autor: Rick Riordan
Edição: 1
Editora: EditoraRiordan
Gênero: Juvenil
Preço de Venda: 2
Preço de Compra: 1
Imposto: 3.5

Qtd: 1
Título: Maldição de Apollo
Autor: Rick Riordan
Edição: 2
Editora: EditoraRiordan
Gênero: Juvenil
Preço de Venda: 2
Preço de Compra: 1
Imposto: 1.5

Ordem de Compra com esse índice não encontrada
```

Figura 15: Teste do Método de Remoção de Ordem de Compra

Ao observar a figura acima, pode-se perceber que o comportamento está como o esperado, visto que, ao passarmos um índice de uma ordem de compra que existe, após o método de remoção, ela já não está mais contida na lista, como indica o resultado do consultar.