
Universidade Federal de Minas Gerais

Escola de Engenharia

ELE078 - Programação Orientada a Objetos

# Documentação TP2

Felipe Bartelt de Assis Pessoa 2016026841

4 de julho de 2022

# Sumário

# 1 Implementação

## 1.1 Classe Date

Para manipular datas, criou-se a classe `Date`, cujos atributos são três `int`s: `ano`, `mes` e `dia`, que armazenam o ano, mês e dia, respectivamente, da data. A classe apresenta construtor *default*, que gera um objeto com a data do dia atual; um construtor por cópia, e um construtor que aceita parâmetros de forma a se definir o ano, mês e dia da data.

Implementou-se o método `elapsed_days()`, que retorna o total de dias decorridos até a data do objeto, considerando-se cada mês como 30 dias e cada ano como 365 dias. Com base nesse método, foi possível implementar os operadores de soma `operator+(Date&)`, que realiza a soma do total de dias das duas datas, retornando a data resultante; e `operator+(int&)` que realiza a soma do total de dias decorridos do objeto com um valor inteiro de dias, retornando então um objeto `Date`, com a data resultante. Esse último método é utilizado para fazer as penalizações da biblioteca. Também se tornou possível definir a operação de comparação `operator>(Date&)`, que realiza a comparação de duas datas por meio de seus dias corridos.

Tornou-se necessário a criação de um método `days2ymd(int)`, para transformar um total de dias em um objeto `Date` com os valores de dia, mês e ano corretos. Também definiu-se o operador `operator<<(ostream&, Date&)` para facilitar a impressão de valores em tela e o operador de atribuição `operator=(Date&)`.

## 1.2 Classe Usuario

A classe Usuario apresenta quatro atributos do tipo `string`: `nome`, `cpf`, `endereco`, `fone` e um atributo do tipo `Date dataPenalizacao`, conforme requisitado.

A classe não apresenta construtor *default*, tendo somente um construtor que exige todos os atributos do tipo `string`, alocando automaticamente o atributo de `dataPenalizacao` como a data atual, além do construtor por cópia. Definiu-se um método *getter* para cada atributo da classe e somente um *setter* para o atributo `dataPenalizacao`.

Os operadores definidos para a classe foram: `operator==(Usuario&)`, que compara todos os atributos de dois Usuarios, o operador de atribuição `operator=(Usuario&)` e o operador para impressão na tela `operator<<(ostream&, Usuario&)`.

## 1.3 Classe Publicacao

A classe Publicacao foi implementada como uma classe abstrata. Ela contém, como requisitado, dois atributos do tipo `int codPublicacao` e `ano` e dois atributos do tipo `string titulo` e `editora`. Somente foi implementado um construtor, que requisita como argumentos, todos os atributos da clase.

Implementou-se *getters* para todos os atributos da classe. Além disso, definiu-se um método `compare(Publicacao&)` para comparar duas publicações, esse método é um método auxiliar para o operador de comparação.

Suas funções virtuais puras consistem nos métodos `emprestimo()`, `devolucao()` e no operador `operator==(Publicacao&)`. Os dois primeiros servem, respectivamente, para configurar o empréstimo e devolução dos tipos de Publicacao, o último é o operador de comparação que deve ser implementado em cada classe derivada. Essas operações tornam a classe polimórfica e simplificam seu uso.

### 1.3.1 Classe Livro

A classe Livro é uma classe derivada de Publicacao, de forma pública. Ela tem os atributos adicionais: `string autores`, `int qtdeExemplares`. A classe apresenta dois construtores: um exige a definição de todos os atributos, tanto os da classe pai quanto os da classe derivada, o outro, somente não exige a quantidade de exemplares. Implementou-se assim dois novos *getters*, para os novos atributos.

O *overloading* das funções virtuais puras foram feitos da seguinte forma. Método `emprestimo()` se a quantidade de exemplares é maior que zero, decrementa-se de uma unidade `qtdeExemplares`, caso contrário gera-se uma exceção `EmprestimoExcpetion`. O método `devolucao()` aumenta em 1 a quantidade de Exemplares. O operador de comparação realiza um `dynamic_cast` na Publicacao comparada para a classe Livro, caso seja retornado um ponteiro nulo, a Publicacao não é um Livro e, portanto não são iguais, retornando-se *false*. Caso a Publicacao seja de fato um Livro, então retorna-se a operação *AND* entre a comparação entre os autores de ambos os Livros, com o resultado do método `compare()`, definido na classe pai.

### 1.3.2 Classe Periodico

A classe Periodico é uma classe derivada de Publicacao, de forma pública. Ela tem os atributos adicionais: `string mes`, `int numEdicao`. A classe apresenta somente um construtor, que exige a definição de todos os atributos, tanto os da classe pai quanto os da classe derivada. Implementou-se assim dois novos *getters*, para os novos atributos.

O *overloading* das funções virtuais puras foram feitos da seguinte forma. Tanto para o método `emprestimo()`, quanto para `devolucao()`, geram-se exceções dos tipos `EmprestimoExcpetion` e `DevolucaoExcpetion`, respectivamente. O operador de comparação realiza um `dynamic_cast` na Publicacao comparada para a classe Periodico, caso seja retornado um ponteiro nulo, a Publicacao não é um Periodico e, portanto não são iguais, retornando-se *false*. Caso a Publicacao seja de fato um Periodico, então retorna-se a operação *AND* entre a comparação entre os meses e número de edição de ambos os Periodicos, com o resultado do método `compare()`, definido na classe pai.

## 1.4 Classe ItemEmprestimo

A classe tem somente um atributo privado do tipo `Date dataDevolucao` e um atributo público `Livro livro`. Crio-se um construtor por cópia e um construtor que tem como parâmetro o Livro referente somente, atribuindo-se automaticamente a data de devolução como um valor nulo, considerado como dia, mês e ano todos com valor 0. Definiu-se então um *setter* para a data de devolução.

Definiu-se três operadores para a classe, `operator==(ItemEmprestimo&)`, que compara a data de devolução e o livro atribuído; `operator==(Livro&)` que compara o livro referente ao ItemEmprestimo com outro livro; e o operator de impressão na tela `oprator<<`.

## 1.5 Classe Emprestimo

A classe apresenta seis atributos privados: `int numero`, `Date dataEmprestimo`, `Date dataPrevDevolucao`, `Usuario *usuario`, `vector<ItemEmprestimo> itens` e `static int proximoNumero`, inicializado como 0. O construtor da classe exige um valor de dataEmprestimo e um ponteiro Usuario, os demais atributos são completados automaticamente pelo construtor.

Implementou-se os métodos de devolução, empréstimo e remoção de um único livro, além do método para devolver todos os livros, conforme requisitado. Ainda, os métodos de adicionar

e remover um objeto ItemEmprestimo do vetor `itens`. Implementou-se também *getters* para todos os atributos da classe, além dos operadores `oprator=`, `oprator==` e `oprator<<`.

O método de adicionar itens ao empréstimo, checa se a data corrente é menor que a data de penalização do usuário, caso positivo, gera-se um `EmprestimoException` e, caso contrário, o item é adicionado.

## 1.6 Classe Biblioteca

A classe Biblioteca gerencia a maioria das funções desejadas. Assim, ela tem três atributos privados que são vetores de ponteiros para Usuario, Publicacao e Emprestimo. A classe somente apresenta o construtor *default*, seu destrutor deleta todos os elementos dinamicamente alocados para formar os vetores de seus atributos. Implementou-se *getters* para todos os seus parâmetros.

As funções para adicionar um usuário e uma publicação à biblioteca simplesmente recebem ponteiros dos respectivos objetos e os adicionam ao vetor correspondente.

O método de adicionar um empréstimo realiza a checagem se o usuário possui uma penalização por meio de seu atributo dataPenalizacao. Se a data for maior que a data atual, então o usuário não pode alugar um livro e um `EmprestimoException` é gerado. Ainda, o método itera por todos os Emprestimos existentes e checa se há algum outro Emprestimo vinculado ao mesmo usuário, cuja data prevista de devolução é menor que a data atual, ou seja, o usuário deveria ter devolvido um livro antes de alugar outro. Nesse caso, também se gera um `EmprestimoException`. Caso não haja nenhum impecilho, o empréstimo é adicionado ao vetor de Emprestimos.

O método para adicionar um ItemEmprestimo a um Emprestimo existente somente checa se o usuário tem outro Emprestimo, cuja devolução já deveria ter sido realizada. Isso, pois o método utiliza a função `add_item` da classe Emprestimo, que já faz a checagem quanto à data de penalização do usuário. Caso não haja problemas, esse método diminui em uma unidade a quantidade de exemplares do Livro relativo ao ItemEmprestimo, além de adicionar o item ao Emprestimo requisitado.

O método para excluir um usuário somente o exclui caso o usuário não tenha nenhum Emprestimo vinculado.

O método para excluir publicações somente checa se a Publicacao é um Livro, caso contrário, a Publicacao é imediatamente removida, no caso positivo, faz-se a checagem se há algum Emprestimo cujos itens vinculados incluam o Livro em questão. Se houver algum empréstimo do Livro, então gera-se uma exceção `RemocaoException`.

O método para remover empréstimos apenas checa se o vetor `itens` é vazio, se sim, o Emprestimo é excluído, se não um `RemocaoException` é gerado. Além disso, ela checa se a devolução já deveria ter sido efetuada, em caso positivo, altera-se a data de penalização do usuário vinculado para uma data 3 dias após o dia atual.

O método para remover um ItemEmprestimo de um Emprestimo utiliza o método da classe Emprestimo para sua remoção. O método também atualiza a data de penalização do usuário, caso necessário. E ainda, se com a remoção do item, o vetor de itens do Emprestimo se torne vazio, esse Emprestimo também é deletado. O método `remove_item_emprestimo` também controla a devolução do objeto Livro, por meio dos métodos dessa classe.

Os métodos `devolve_livro` e `devolve_todos_livro` utilizam o método `remove_item_emprestimo` para realizar a devolução dos Livros. Ainda, ao devolver todos os livros, o segundo método também chama a função `remove_emprestimo`. Os métodos chamam seus métodos análogos da classe Emprestimo, que tem os mesmos nomes.

Para as pesquisas por título e autor, utilizou-se regex para simplificar as pesquisas por substrings. Assim, a `string` a ser pesquisada é transformada em uma expressão regular e então itera-se todas as publicações, checando se há alguma ocorrência do padrão definido no título

ou autor da mesma. Ressalta-se que essa pesquisa é *case-sensitive*. Assim, o funcionamento de `pesquisa_titulo` e `pesquisa_autor` é análogo, apesar do primeiro retornar um vetor de ponteiros para Publicacao, enquanto o segundo retorna um vetor de Livro.

### 1.7 Classe Interface

A classe Interface tem dois atributos `Biblioteca biblioteca` e o texto padrão do menu da interface `static const string TEXTO_MENU`. O único construtor da classe exige uma Biblioteca como argumento.

O menu da interface é obtido pelo método `menu()`, que apresenta um switch com todas as optções de função que podem ser realizadas. O usuário digita o caracter referente ao comando desejado, segundo o menu impresso em tela, para selecionar a função desejada. Como *default*, imprime-se novamente o menu. Para sair da interface, o caracter escolhido é a tecla `Q`. As funções exigidas foram então implementadas em ordem, sendo seus respectivos comandos as teclas alfabéticas do teclado em ordem, a partir de `E`. São utilizadas os métodos implementados na Biblioteca para realizar as funções escolhidas. O método retorna um valor booleano que indica se a tecla `Q` foi apertada ou não. Todas as chamadas para os métodos de Biblioteca estão entre cláusulas `try` e `catch`, de forma a imprimir em tela os erros, evitando a interrupção do programa.

O método `main()`, que efetivamente implementa a interface, simplesmente implementa um loop infinito que executa o método `menu()`, até que a tecla `Q` seja apertada.

Grande parte dos métodos de Biblioteca necessitam da passagem de um usuário, empréstimo, publicação ou item de empréstimo como argumento de função. Assim, para que isso fosse implementado numa interface, tornou-se necessário a impressão de todos os objetos desse tipo em tela. Assim o usuário, seleciona o índice de referência do objeto para o qual deseja realizar uma função. Esses índices são majoritariamente a ordem dos vetores, com exceção das Publicações, que são selecionadas com base no seu código de publicação.

### 1.8 Classes de erro

Criou-se a classe abstrata `BibliotecaException` para o tratamento de erros da Bilbioteca. Dela foram derivadas três subclasses `EmprestimoException`, que se refere a erros relacionados a funções de empréstimo, `DevolucaoException`, relacionada a funções com relação às devoluções e `RemocaoException`, referente a funções que tentam remover objetos da Biblioteca.

## 2 Testes

O arquivo `main.c` apresenta um script de testes para cada função e exceção da classe Biblioteca, por meio do qual nota-se o correto funcionamento de todas as funções implementadas. Ainda, para se testar a classe de interface, basta tomar `const int TEST_WITH = 0` no início do arquivo. Essa variável controla se deve se executar uma interface ou uma sequência de testes. Caso a Interface seja testada, note a dependência da iteratividade com o usuário por meio do *prompt*. Ainda, quanto a Interface, utilizou-se comandos `system("clear")` que podem não funcionar em todos os *shells* e sistemas operacionais.

## 3 Especificações Utilizadas

As versões do *vscode*, g++, gcc, gdb e as especificações do computador utiizado são mostradas abaixo, conforme as saídas dos comandos utilizados no terminal.

```
>> g++ --version
g++ (GCC) 12.1.0

>> gcc --version
gcc (GCC) 12.1.0

>> gdb -v
GNU gdb (GDB) 12.1

>> code --version
1.68.1
30d9c6cd9483b2cc586687151bcbcd635f373630
x64

>> neofetch
                    /-                      fbartelt@fbartelt-nitroan51555
                   ooo:                     ------------------------------
                  yoooo/                    OS: ArcoLinux
                 yooooooo                   Kernel: 5.18.5-arch1-1
                yoooooooooo                 Uptime: 10 hours, 12 mins
               yooooooooooo                 Packages: 1216 (pacman)
             .yooooooooooooo                Shell: zsh 5.9
            .ooooooooooooooo                Resolution: 1920x1080
           .ooooooarcooooooooo              WM: i3
          .ooooooooo-oooooooooo             Theme: Arc-Dark [GTK2/3]
         .ooooooooo-  ooooooooooo           Icons: DarK-svg [GTK2/3]
        :ooooooooo.     :ooooooooo          Terminal: kitty
       :ooooooooo.       :ooooooooo         CPU: Intel i5-10300H (8) @ 4.500GHz
      :oooarcooo           .oooarcooo       GPU: Intel CometLake-H GT2 [UHD Graphics]
     :ooooooooy             .ooooooooo      GPU: NVIDIA GeForce GTX 1650 Mobile / Max-Q
    :ooooooooo    /ooooooooooooooooooo      Memory: 4271MiB / 7779MiB (54%)
   :ooooooooo       .-ooooooooooooooooo.
  oooooooo-              -ooooooooooooo.
 ooooooooo-                  .-ooooooooo.
oooooooo.                       -ooooooooo
```