Universidade Federal de Juiz de Fora
Pós-Graduação em Modelagem Computacional
Engenharia de Software

Eduardo Santos de Oliveira Marques

Lista de Exercícios 2

Juiz de Fora
2023

Questão 1. Escolha um dos cenários descritos nos exercícios 1 e 2 da lista anterior e implemente-o em C++ utilizando os conceitos de classe, herança, polimorfismo, classe abstrata, entre outros que forem necessários.

O código deve ser desenvolvido levando em conta apenas a estrutura do diagrama de classes, isto é, não é preciso implementar as funcionalidades propostas no diagrama.

Para facilitar o entendimento e verificar se a implementação está correta sugere-se que a impressão de mensagens.

Atenção e cuidado com: alocação dinâmica de memória. Implementar o construtor de cópia e operador de atribuição se suas classes utilizarem alocação dinâmica de memória. Liberar toda memória alocada de forma dinâmica no destrutor das respectivas classes.

Resolução:

Abaixo é feita a implementação do Sistema de Gerenciamento de Biblioteca em C++, levando em consideração a estrutura do diagrama de classes.

```
#include <iostream>
#include <string>
#include <vector>

// Classe Livro
class Livro {
private:
    std::string titulo;
    std::string autor;
    std::string ISBN;

public:
    Livro(const std::string& _titulo, const std::string& _autor, const std::string& _ISBN)
        : titulo(_titulo), autor(_autor), ISBN(_ISBN) {}

    std::string getTitulo() const { return titulo; }
    std::string getAutor() const { return autor; }
    std::string getISBN() const { return ISBN; }
};

// Classe Usuario
class Usuario {
private:
    std::string nome;
    std::string email;
    std::string senha;

public:
    Usuario(const std::string& _nome, const std::string& _email, const std::string& _senha)
        : nome(_nome), email(_email), senha(_senha) {}

    std::string getNome() const { return nome; }
    std::string getEmail() const { return email; }

    void fazerEmprestimo(Livro& livro);
```

```
    void fazerReserva(Livro& livro);
};

void Usuario::fazerEmprestimo(Livro& livro) {
    std::cout << "Usuario " << nome << " fez um emprestimo do livro \"" << livro.getTitulo()
}

void Usuario::fazerReserva(Livro& livro) {
    std::cout << "Usuario " << nome << " fez uma reserva do livro \"" << livro.getTitulo() <<
}

// Classe Biblioteca
class Biblioteca {
private:
    std::string nome;
    std::string endereco;
    std::vector<Livro> acervo;

public:
    Biblioteca(const std::string& _nome, const std::string& _endereco)
        : nome(_nome), endereco(_endereco) {}

    void adicionarLivro(const Livro& livro);
    void removerLivro(const Livro& livro);
    void listarLivros() const;
};

void Biblioteca::adicionarLivro(const Livro& livro) {
    acervo.push_back(livro);
}

void Biblioteca::removerLivro(const Livro& livro) {
    for (auto it = acervo.begin(); it != acervo.end(); ++it) {
        if (it->getISBN() == livro.getISBN()) {
            acervo.erase(it);
            break;
        }
    }
}

void Biblioteca::listarLivros() const {
    std::cout << "Acervo da biblioteca \"" << nome << "\":" << std::endl;
    for (const auto& livro : acervo) {
        std::cout << "Titulo: " << livro.getTitulo() << std::endl;
        std::cout << "Autor: " << livro.getAutor() << std::endl;
        std::cout << "ISBN: " << livro.getISBN() << std::endl;
        std::cout << "----------" << std::endl;
    }
}

// Classe Emprestimo
class Emprestimo {
private:
    std::string dataEmprestimo;
    std::string dataDevolucao;
    bool status;
```

```
public:


 Emprestimo(const std::string& _dataEmprestimo, const std::string& _dataDevolucao)
        : dataEmprestimo(_dataEmprestimo), dataDevolucao(_dataDevolucao), status(true) {}

    void devolver();
};

void Emprestimo::devolver() {
    status = false;
    std::cout << "Emprestimo devolvido" << std::endl;
}

// Classe Reserva
class Reserva {
private:
    std::string dataReserva;
    bool status;

public:
    Reserva(const std::string& _dataReserva)
        : dataReserva(_dataReserva), status(true) {}

    void cancelar();
};

void Reserva::cancelar() {
    status = false;
    std::cout << "Reserva cancelada" << std::endl;
}

int main() {
    // Criando objetos do cenario da biblioteca
    Biblioteca biblioteca("Biblioteca Central", "Rua Principal, 123");

    Livro livro1("Aprendendo C++", "John Smith", "9781234567890");
    Livro livro2("Introducao a Programacao", "Jane Johnson", "9780987654321");
    Livro livro3("Algoritmos e Estruturas de Dados", "Robert Roberts", "9789876543210");

    biblioteca.adicionarLivro(livro1);
    biblioteca.adicionarLivro(livro2);
    biblioteca.adicionarLivro(livro3);

    // Listando os livros disponiveis na biblioteca
    biblioteca.listarLivros();

    // Criando usuarios
    Usuario usuario1("Joao", "joao@example.com", "123456");
    Usuario usuario2("Maria", "maria@example.com", "abcdef");

    // Realizando emprestimos e reservas
    usuario1.fazerEmprestimo(livro1);
    usuario2.fazerReserva(livro2);

    return 0;
}
```

Neste exemplo, as classes *Livro*, *Usuario*, *Biblioteca*, *Emprestimo* e *Reserva* foram implementadas de acordo com a estrutura do diagrama de classes. Algumas funcionalidades como *adicionarLivro*, *removerLivro*, *listarLivros*, *fazerEmprestimo* e *fazerReserva* foram implementadas com mensagens de saída para facilitar o entendimento e verificar se a implementação está correta. É importante ressaltar que a implementação das funcionalidades reais (como a persistência de dados, validação de empréstimos e reservas, por exemplo) não foi realizada nesse exemplo, uma vez que o foco era demonstrar a estrutura das classes e a interação entre elas.

O código foi desenvolvido e testado na plataforma `https://www.onlinegdb.com/`. Ele pode ser acessado através do link: `https://onlinegdb.com/3I3cHPycD`.