UFPB - Centro de Informática

POO - 2024.2 - Prof. Carlos Eduardo Batista

1) Você foi contratado para desenvolver um sistema para um hotel. O hotel possui uma variedade de quartos que podem ser reservados pelos hóspedes. Cada quarto possui informações como número, tipo (standard, luxo, suíte), capacidade e preço da diária. O hotel deseja acompanhar o histórico de reservas de cada quarto e os hóspedes que já os reservaram. Cada hóspede que faz uma reserva deve fornecer seu nome, CPF e telefone. O sistema deve permitir:

- Realizar uma nova reserva, onde um hóspede escolhe um quarto disponível e informa quantos dias deseja reservar.
- O valor total da reserva é calculado com base no preço da diária e na quantidade de dias.
- Exibir um histórico dos hóspedes que reservaram cada quarto, incluindo as datas das reservas.

Considere que um hóspede chamado João Silva, CPF 987.654.321-00, deseja reservar um quarto específico do hotel, uma suíte com capacidade para 4 pessoas, número 101, e preço da diária de R$ 300,00. João pretende reservar este quarto por 3 dias. Com base na história e no cenário, responda às seguintes perguntas:

a. Identifique as classes principais que você criaria para representar esse sistema.
b. Descreva como seriam criados os objetos manipulados nesse cenário.
c. Liste atributos e métodos que cada classe teria, e explique o encapsulamento deles.
d. Explique os tipos de associações que ocorrem entre as classes.
e. Esboce as classes em C++ com os atributos e métodos mencionados.

2) Explique a saída do código, e como os construtores e destrutores foram utilizados

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

class sala {
private:
    std::string nome;
    int capacidade;
public:
    sala(const std::string& n, int c) : nome(n),
capacidade(c) {
        std::cout<<"sala("<<nome<<")\n";
    }
    ~sala() {
        std::cout<<"~sala("<<nome<<")\n";
    }
    void mostrar() const {
        std::cout<<"Sala: "<<nome<<" ("<<
capacidade << ")\n";
    }
};

class filme {
private:
    std::string titulo;
    int ano;
    sala* sala_filme;
public:
    filme(const std::string& t, int a, sala* s)
        : titulo(t), ano(a), sala_filme(s) {
        std::cout<<"filme("<<titulo<<")\n";
    }
    ~filme() {
        std::cout<<"~filme("<<titulo<<")\n";
    }
    void mostrar() const {
        std::cout << "Filme: " << titulo << " ("
<< ano << ")\n";
        sala_filme->mostrar();
    }
};
```

```
class sessao {
private:
    std::string horario;
    sala* sala_sessao;
public:
    sessao(const std::string& h, sala* s)
        : horario(h), sala_sessao(s) {
        std::cout<<"sessao("<<horario<<")\n";
    }
    ~sessao() {
        delete sala_sessao;
        std::cout<<"~sessao("<<horario<<")\n";
    }
    void mostrar() const {
        std::cout << "Sessão: " << horario <<
'\n';
        sala_sessao->mostrar();
    }
};

int main() {
    std::string nome_sala_1 = "Sala 1";
    sala* sala_filme_1 = new sala(nome_sala_1,
100);
    nome_sala_1 = "Sala VIP";

    filme filme_1("Interstellar", 2014,
sala_filme_1);

    std::string horario_sessao_2 = "14:00";
    sessao sessao_1(horario_sessao_2, new
sala("Sala 2", 150));
    horario_sessao_2 = "16:30";
    std::cout << "\nDados do filme:\n";
    filme_1.mostrar();
    std::cout << "\nDados da sessão:\n";
    sessao_1.mostrar();
    delete sala_filme_1;
    return 0;
}
```

3) Analise o seguinte código C++ que implementa um sistema básico de contas bancárias:

```
#include <iostream>
using namespace std;

class conta_bancaria {
private:
    string numero_conta;
protected:
    double saldo;
    void registrar_operacao(string tipo, double
valor) {
        cout << "Operacao " << tipo << ": R$ "
<< valor << endl;
        cout << "Saldo atual: R$ " << saldo <<
endl;
    }
public:
    conta_bancaria(const string& num, double
inicial)
        : numero_conta(num), saldo(inicial) {
        cout << "Conta " << numero_conta << "
criada com saldo R$ " << saldo << endl;
    }

    void consultar_saldo() {
        cout << "Conta " << numero_conta <<
endl;
        cout << "Saldo: R$ " << saldo << endl;
    }
};

class conta_poupanca : public conta_bancaria {
private:
    double taxa_juros;
public:
    conta_poupanca(const string& num, double
inicial, double taxa)
        : conta_bancaria(num, inicial),
taxa_juros(taxa) {
    }

    void aplicar_juros() {
        // Implementar cálculo e aplicação dos
juros
        // Use o atributo protegido saldo e o
método protegido registrar_operacao
    }
};
```

```
class conta_corrente : public conta_bancaria {
private:
    double limite;
public:
    conta_corrente(const string& num, double
inicial, double lim)
        : conta_bancaria(num, inicial),
limite(lim) {
    }

    bool sacar(double valor) {
        // Implementar saque considerando o
limite
        // Use o atributo protegido saldo e o
método protegido registrar_operacao
        // Retornar true se o saque for
possível e false caso contrário
    }
};

int main() {
    conta_poupanca cp("001", 1000, 0.5);
    conta_corrente cc("002", 2000, 500);

    cp.consultar_saldo();
    cp.aplicar_juros();
    cp.consultar_saldo();

    cc.consultar_saldo();
    if(cc.sacar(2300))
        cout << "Saque realizado com sucesso"
<< endl;
    else
        cout << "Saldo insuficiente" << endl;
    cc.consultar_saldo();

    return 0;
}
```

a) Por que o atributo saldo e o método registrar_operacao foram declarados como protected? Qual a vantagem disso neste contexto de herança?

b) Complete a implementação do método aplicar_juros() da classe conta_poupanca. Os juros devem ser calculados como saldo * (taxa_juros/100) e somados ao saldo atual. Use o método protegido registrar_operacao para registrar esta operação.

c) Complete a implementação do método sacar() da classe conta_corrente. O saque só deve ser permitido se o valor não exceder o saldo + limite. Use o método protegido registrar_operacao para registrar esta operação quando for bem sucedida.

d) Modifique a classe conta_corrente para adicionar um método protegido verificar_limite() que retorna true se um determinado valor está dentro do limite disponível. Use este método dentro de sacar().

e) Qual será a saída exata deste programa? Justifique a ordem de chamada dos construtores e destrutores.