UFPB - Centro de Informática

Primeira Prova - POO - 2024.2 - Prof. Carlos Eduardo Batista

1) (5,0) Você foi contratado para desenvolver um sistema de gestão para um parque de diversões. O parque possui diversas atrações que podem ser acessadas pelos visitantes mediante compra de ingressos. Cada atração possui informações como nome, categoria (infantil, família, radical), capacidade máxima, altura mínima e tempo de duração. O parque deseja acompanhar o histórico de visitação de cada atração e os visitantes que as utilizaram. Cada visitante que compra um ingresso deve fornecer seu nome, CPF e idade. O sistema deve permitir:

- Emitir um ingresso para uma atração, verificando se o visitante atende aos requisitos.
- Calcular o preço do ingresso com base na categoria da atração e possíveis descontos.
- Exibir relatórios de visitação de cada atração, incluindo horários de pico e perfil dos visitantes

Um visitante chamado Darci Oliveira, CPF 123.456.789-00, 12 anos, deseja comprar um ingresso para a montanha-russa "Tornado Veloz", classificada como radical, com altura mínima de 1,40m, duração média de 3 minutos e preço base de R$ 50,00. Darci tem 1,45m de altura. Com base nesse cenário, responda:

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) (2,0) Explique a saída do código abaixo, e como os construtores e destrutores foram utilizados:

```
#include <iostream>
#include <string>
class veiculo {
private:
    std::string modelo;
    int ano;
public:
    veiculo(const std::string& m, int a) :
modelo(m), ano(a) {
        std::cout<<"veiculo("<<modelo<<")\n";
    }
    ~veiculo() {
        std::cout<<"~veiculo("<<modelo<<")\n";
    }
    void mostrar() const {
        std::cout<<"Veículo: "<<modelo<<"
("<<ano<<")\n";
    }
};
class corrida {
private:
    std::string origem;
    veiculo* veiculo_corrida;
public:
    corrida(const std::string& o, veiculo* v)
        : origem(o), veiculo_corrida(v) {
        std::cout<<"corrida("<<origem<<")\n";
    }
    ~corrida() {
        delete veiculo_corrida;
        std::cout<<"~corrida("<<origem<<")\n";
    }
    void mostrar() const {
        std::cout<<"Origem: "<<origem<<"\n";
        veiculo_corrida->mostrar();
    }
};
int main() {
    std::string modelo = "Sedan";
    veiculo* carro = new veiculo(modelo,
2022);
    modelo = "SUV";

    corrida viagem("Centro", carro);
    std::cout<<"\nDetalhes:\n";
    viagem.mostrar();

    return 0;
}
```

3. (3,0) Responda às seguintes questões sobre os conceitos fundamentais de Programação Orientada a Objetos:

a) Explique as diferenças entre os modificadores de acesso private, protected e public em C++, especialmente no contexto de herança. Dê exemplos de atributos que deveriam ser declarados com cada um desses modificadores.

b) Compare as abordagens de reuso de código por herança (relação "é um") versus composição (relação "tem um"). Para cada um dos cenários abaixo, indique qual abordagem seria mais adequada e justifique sua escolha:

- Um sistema de veículos onde temos carros, motos e caminhões
- Um sistema onde um smartphone possui câmera e bateria
- Um sistema bancário com contas correntes e contas poupança