UFPB - Centro de Informática
Segunda Prova - POO - 2024.2 - Prof. Carlos Eduardo Batista

1) (2,5) Um sistema de monitoramento para usinas de energia renovável possui diferentes tipos de sensores.

a) (1,25) Explique as diferenças entre herança pública, protegida e privada em C++. Qual seria mais adequada para implementar os sensores desse sistema? Justifique.

b) (1,25) Complete o código abaixo adicionando um método estático `registrar_sensor()` e um atributo estático `total_sensores` na classe `sensor_base`.

```
class sensor_base {
private:
    // Adicione o atributo estático
public:
    // Adicione o método estático
};
```

2) (2,5) Analise o código abaixo de um sistema para gerenciamento de algoritmos de IA:

```
class algoritmo {
protected:
    std::string nome;
    int complexidade;
public:
    algoritmo(std::string n, int c) : nome(n), complexidade(c) {}
    virtual double calcular_desempenho() { return 0.0; }
    virtual ~algoritmo() {}
};
class redes_neurais : public algoritmo {
public:
    double calcular_desempenho() override {
        return /* cálculo específico */;
    }
};
```

a) (1,0) O que aconteceria se removêssemos a palavra-chave `virtual` do método `calcular_desempenho()` na classe base? Como isso afetaria o uso de polimorfismo?

b) (1,0) Altere a classe algoritmo (torná-la abstrata) com o método `calcular_desempenho()` como virtual puro e adicione um método virtual puro `otimizar()`.

3) (2,5) Complete o código abaixo para um sistema de processamento de sinais digitais, e em seguida detalhe sua saída:

```
class processador_sinal {
    ____ int instancias_ativas;
// a) Complete para contar instâncias
public:
    processador_sinal() {
instancias_ativas++; }
    virtual ~processador_sinal() {
instancias_ativas--; }
    ____ void filtrar() ____
// b) Complete para método virtual puro
    ____ int get_instancias() { ____ }
// c) Complete para obter instâncias
};

class filtro_passa_baixa : ____
processador_sinal {
// d) Complete herança
public:
    void filtrar() override {
        std::cout << "Filtrando
frequências altas" << std::endl;
    }
};

processador_sinal* p1 = new
filtro_passa_baixa();
p1->filtrar();
std::cout << "Instâncias: " <<
filtro_passa_baixa::get_instancias() <<
std::endl;
delete p1;

// e) Detalhe a saída da execução
```

4) (2,5) Considere um sistema de criptografia com diferentes algoritmos de cifragem:

```
class cifrador {
public:
    virtual std::string cifrar(const std::string& texto) = 0;
    virtual std::string decifrar(const std::string& texto_cifrado) = 0;
    virtual ~cifrador() {}
};

class cifra_cesar : public cifrador {
private:
    int deslocamento;
public:
    std::string cifrar(const std::string& texto) override {
        // implementação resumida
    }
};
```

a) (0,5) Por que a classe cifrador é uma classe abstrata? Quais são as características que definem uma classe abstrata em C++?
b) (1,5) Esboce um sistema de gerenciamento de cifradores usando o padrão *singleton* que permita registrar diferentes algoritmos.
c) (0,5) Como você poderia modificar esse sistema para permitir a combinação de diferentes algoritmos de cifragem em sequência? Esboce o código.