Respostas Instrução Prática Java – PO-P01 – Parte Teórica

Pesquise no Google e responda às perguntas abaixo.

1. O que é uma classe em Java e qual é a diferença entre uma classe e um objeto? Dê 5 exemplos mostrando-os em C++ e em Java.

Em Java e C++, uma classe é uma estrutura que define um tipo de objeto. Ela serve como um modelo ou plano para criar objetos que compartilham características comuns. Uma classe contém atributos (variáveis) e métodos (funções) que descrevem o comportamento e os dados que os objetos dessa classe podem ter.

Por outro lado, um objeto é uma instância de uma classe, ou seja, é criado a partir da classe e representa uma entidade específica desse tipo. Por exemplo, se a classe for Carro, um objeto seria um carro específico com características únicas, como cor, modelo e marca.

Exemplos em c++:

```
// Classe Ponto
class Ponto {
public:
   int x, y;
   Ponto(int x, int y) : x(x), y(y) {}
};

// Objeto ponto1 da classe Ponto
Ponto ponto1(3, 5);
```

```
// Classe Círculo
class Circulo {
public:
   double raio;
   Circulo(double r) : raio(r) {}
};

// Objeto circulo1 da classe Circulo
Circulo circulo1(7.0);
```

Exemplos em Java:

```
// Classe Livro
class Livro {
   String titulo;
   int anoPublicacao;
   Livro(String titulo, int ano) {
      this.titulo = titulo;
      this.anoPublicacao = ano;
   }
}

// Objeto livro1 da classe Livro
Livro livro1 = new Livro("Dom Casmurro", 1899);
```

```
// Classe Retângulo
class Retangulo {
   double comprimento, largura;
   Retangulo(double c, double l) {
      this.comprimento = c;
      this.largura = l;
   }
}

// Objeto retangulo1 da classe Retangulo
Retangulo retangulo1 = new Retangulo(5.0, 3.0);
```

*Estes exemplos mostram como você pode criar classes que representam tipos de objetos (como **Ponto**, **Círculo**, **Livro** e **Retângulo**) e depois instanciar objetos específicos (**ponto1**, **circulo1**, **livro1** e **retangulo1**) dessas classes. As classes definem a estrutura e comportamento, enquanto os objetos são instâncias específicas que utilizam essa estrutura para representar entidades distintas.*

2. Como você declara uma variável em Java e quais são os tipos de dados primitivos mais comuns? Faça um paralelo entre isso e a mesma coisa na linguagem C++

Em java, para declarar uma variável, você utiliza a seguinte sintaxe:

tipoDaVariavel nomeDaVariavel;

Ex:

int numero;

String nome;

double valor;

Os tipos de dados primitivos mais comuns em Java são:

1. **int:** Representa números inteiros.
2. **double:** Armazena números de ponto flutuante de dupla precisão.
3. **boolean:** Pode conter apenas **true** ou **false**.
4. **char:** Armazena um único caractere Unicode.
5. **byte:** Armazena números inteiros pequenos.
6. **short:** Armazena números inteiros um pouco maiores que o byte.
7. **long:** Para armazenar números inteiros longos.

Em C++, a declaração de variáveis segue uma sintaxe semelhante:

tipoDaVariavel nomeDaVariavel;

Ex:

int numero;

std::string nome;

double valor;

No entanto, C++ possui alguns tipos de dados primitivos diferentes:

1. **int:** Assim como em Java, representa números inteiros.
2. **double:** Similar ao Java, armazena números de ponto flutuante de dupla precisão.
3. **bool:** Correspondente ao tipo booleano, armazena **true** ou **false**.
4. **char:** Armazena um único caractere, mas em C++ não é necessariamente Unicode.
5. **short:** Armazena números inteiros curtos.
6. **long:** Em C++, o tamanho pode variar entre sistemas, mas normalmente é usado para números inteiros longos.
7. **long long:** Utilizado para números inteiros muito longos.

Ambas as linguagens possuem tipos de dados primitivos semelhantes, como inteiros, ponto flutuante, booleanos e caracteres, mas os nomes e nuances específicas podem variar entre elas.

3. Explique o conceito de herança em Java e como você pode criar uma subclasse a partir de uma classe existente. Faça um paralelo com C++, apresentando 5 exemplos.

A herança em Java (assim como em C++) é um dos pilares da programação orientada a objetos, permitindo que uma classe herde características (métodos e variáveis) de outra classe. Isso significa que uma nova classe (conhecida como subclasse ou classe derivada) pode estender e reutilizar o comportamento e os atributos de uma classe existente (chamada de superclasse ou classe base).

*Para criar uma subclasse em Java a partir de uma classe existente, você usa a palavra-chave **extends**.*

Exemplos em Java:

```
// Classe base (superclasse) Veiculo
class Veiculo {
    String marca;
    int ano;
    Veiculo(String marca, int ano) {
        this.marca = marca;
        this.ano = ano;
    }
    void mostrarDetalhes() {
        System.out.println("Marca: " + marca + ", Ano: " + ano);
    }
}

// Subclasse Carro que herda de Veiculo
class Carro extends Veiculo {
    int numPortas;
    Carro(String marca, int ano, int numPortas) {
        super(marca, ano);
        this.numPortas = numPortas;
    }
    void mostrarDetalhesCarro() {
        mostrarDetalhes(); // Chama o método da superclasse
        System.out.println("Número de portas: " + numPortas);
    }
}

// Criando um objeto da subclasse Carro
Carro meuCarro = new Carro("Toyota", 2022, 4);
meuCarro.mostrarDetalhesCarro();
```

Em C++:

```
// Classe base (superclasse) Veiculo
class Veiculo {
protected:
```

```
    std::string marca;
    int ano;
public:
    Veiculo(std::string marca, int ano) : marca(marca), ano(ano) {}
    void mostrarDetalhes() {
        std::cout << "Marca: " << marca << ", Ano: " << ano << std::endl;
    }
};

// Subclasse Carro que herda de Veiculo
class Carro : public Veiculo {
private:
    int numPortas;
public:
    Carro(std::string marca, int ano, int numPortas) : Veiculo(marca, ano), numPortas(numPortas) {}
    void mostrarDetalhesCarro() {
        mostrarDetalhes(); // Chama o método da superclasse
        std::cout << "Número de portas: " << numPortas << std::endl;
    }
};

// Criando um objeto da subclasse Carro
Carro meuCarro("Toyota", 2022, 4);
meuCarro.mostrarDetalhesCarro();
```

*Nos exemplos acima, a classe **Carro** herda da classe **Veiculo** tanto em Java quanto em C++. Isso permite à subclasse **Carro** acessar os atributos e métodos da superclasse **Veiculo**. Em Java, a palavra-chave **extends** é usada para estabelecer a herança, enquanto em C++, **public** é usado para indicar a visibilidade dos membros da classe base para a classe derivada.*

4. Quando declaramos uma variável em Java, temos, na verdade, um ponteiro. Em C++ é diferente. Discorra sobre esse aspecto.

Em Java, todas as variáveis de tipos de objeto são referências ou ponteiros indiretos para os objetos na memória. Quando você declara uma variável de um tipo de objeto em Java, como por exemplo:

```
MinhaClasse objeto;
```

O que está sendo criado é uma referência (ou um ponteiro) para um objeto do tipo **MinhaClasse**. Para realmente criar o objeto, é necessário utilizar o operador **new**:

```
objeto = new MinhaClasse();
```

A variável **objeto** não contém o objeto em si, mas sim um endereço de memória que aponta para onde o objeto está armazenado. Isso permite que você acesse e manipule o objeto por meio dessa referência.

Em C++, a situação é um pouco diferente. Você pode criar variáveis que são objetos diretamente, sem a necessidade de referências ou ponteiros. Por exemplo:

MinhaClasse objeto;

Nesse caso, **objeto** é uma instância real da classe **MinhaClasse**, não é um ponteiro para um objeto. Não há a necessidade de usar **new** para criar o objeto (a menos que você queira alocá-lo dinamicamente).

Essa diferença fundamental entre Java e C++ afeta a maneira como os objetos são tratados e manipulados. Em Java, os objetos são sempre manipulados por referência, enquanto em C++, você pode optar por ter objetos diretamente ou usar ponteiros para eles, dependendo das necessidades do seu programa.