**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.**

Uma classe em Java é uma estrutura que serve como modelo para criar objetos. Ela define atributos (variáveis) e métodos (funções) que representam o comportamento dos objetos. As classes em Java são a base para a programação orientada a objetos.

Diferença entre Classe e Objeto:

Classe: É uma estrutura que define as características comuns de um grupo de objetos. Pode conter atributos e métodos.

Objeto: É uma instância específica de uma classe. É uma entidade real que possui estado (atributos) e comportamento (métodos) definidos pela classe.

Exemplos:

- Classe Pessoa em C++

```cpp
class Pessoa {
public:
    string nome;
    int idade;

    void exibirInformacoes() {
        cout << "Nome: " << nome << ", Idade: " << idade << endl;
    }
};
```

- Classe Pessoa em Java

```java
public class Pessoa {
    public String nome;
    public int idade;

    public void exibirInformacoes() {
        System.out.println("Nome: " + nome + ", Idade: " + idade);
    }

}
```

- Classe Carro em C++

```cpp
class Carro {
public:
    string marca;
    string modelo;
    int ano;

    Carro(string m, string mod, int a) : marca(m), modelo(mod), ano(a) {}

    void exibirDetalhes() {
        cout << "Marca: " << marca << ", Modelo: " << modelo << ", Ano: " << ano << endl;
    }
};
```

- Classe Carro em Java

```java
public class Carro {
    private String marca;
    private String modelo;
    private int ano;

    // Constructor
    public Carro(String m, String mod, int a) {
        marca = m;
        modelo = mod;
        ano = a;
    }

    public void exibirDetalhes() {
        System.out.println("Marca: " + marca + ", Modelo: " + modelo + ", Ano: " + ano);
    }

    public static void main(String[] args) {
        Carro meuCarro = new Carro("Toyota", "Corolla", 2022);
        meuCarro.exibirDetalhes();
    }
}
```

-

- Classe Turma em C++

```cpp
class Turma {
public:
    vector<string> alunos;

    void adicionarAluno(string nome) {
        alunos.push_back(nome);
    }

    void exibirAlunos() {
        cout << "Alunos na turma:" << endl;
        for (const string& aluno : alunos) {
            cout << aluno << endl;
        }
    }
};
```

- Classe Turma em Java

```java
public class Turma {
    private ArrayList<String> alunos = new ArrayList<>();

    public void adicionarAluno(String nome) {
        alunos.add(nome);
    }

    public void exibirAlunos() {
        System.out.println("Alunos na turma:");
        for (String aluno : alunos) {
            System.out.println(aluno);
        }
    }

    public ArrayList<String> getAlunos() {
        return new ArrayList<>(alunos);
    }

    public void setAlunos(ArrayList<String> novosAlunos) {
        alunos = new ArrayList<>(novosAlunos);
    }
```

- Classe Aluno em C++

```cpp
class Aluno {
    private:
        string nome;
        string email;
        int nota1;
        int nota2;
    public:
    ~Aluno(){

    }
    Aluno(){
        nome = "";
        email = "";
        nota1 = 0;
        nota2 = 0;
    }
    Aluno(string _nome, string _email, int _nota1, int _nota2){
        nome = _nome;
        email = _email;
        nota1 = _nota1;
        nota2 = _nota2;
    }
    string getNome();
    string getEmail();
    int getNota1();
    int getNota2();

    void setNome(string _nome);
    void setEmail(string _email);
    void setNota1(int _nota1);
    void setNota2(int _nota2);
};
```

-

- Classe Aluno Em Java

```java
public class Aluno {
    private String nome;
    private String email;
    private int nota1;
    private int nota2;

    public Aluno(String _nome, String _email, int _nota1, int _nota2) {
        nome = _nome;
        email = _email;
        nota1 = _nota1;
        nota2 = _nota2;
    }

    public String getNome() {...}
    public String getEmail() {...}
    public int getNota1() {...}
    public int getNota2() {...}

    public void setNome(String _nome) {
        nome = _nome;
    }
    public void setEmail(String _email) {
        email = _email;
    }
    public void setNota1(int _nota1) {
        nota1 = _nota1;
    }
    public void setNota2(int _nota2) {
        nota2 = _nota2;
    }
}
```

- Classe Data em C++

```cpp
class Data {
    private:
        int dia;
        int mes;
        int ano;
        string msgErro;
    public:
        Data();
        ~Data();
        Data(int dia, int mes, int ano);
        string dataValida(int dia, int mes, int ano);
        string alteraData(int dia, int mes, int ano);
        string to_string_zeros(int numero);
        string dataParaString(string format = "pt-br");
        string dataPorExtenso();
        string getErro();
};
```

- Classe Data em Java

```java
public class Data {
    private int dia;
    private int mes;
    private int ano;
    private String msgErro;

    public Data(int dia, int mes, int ano) {
        this.dia = dia;
        this.mes = mes;
        this.ano = ano;
    }

    public String dataValida(int dia, int mes, int ano) {...

    public String alteraData(int dia, int mes, int ano) {...

    private String to_string_zeros(int numero) {...

    public String dataParaString(String format) {...

    public String dataPorExtenso() {...

    public String getErro() {...
```

**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++**
Uma variavel Java é declarada especificando primeiro o tipo de dado seguido do nome  da variavel

```
    String nome;
    int idade;
    double salario;
```

**int**: Armazena números inteiros.
**double**: Armazena números de ponto flutuante (decimais).
**char**: Armazena um único caractere.
**boolean**: Armazena valores booleanos (verdadeiro ou falso).
**byte, short, long, float**: Outros tipos numéricos que armazenam números inteiros ou de ponto flutuante com diferentes faixas de valores.

Os tipos de dados primitivos em C++ são semelhantes.

**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.**

Herança é permite que uma classe herde os atributos e métodos de outra classe. Isso facilita a reutilização de código e permite a criação de hierarquias de classes. Em Java, você usa a palavra-chave `extends` para criar uma subclasse.

**Exemplo em Java:**

```java
// Superclasse (classe base)
class Animal {
    void fazerSom() {
        System.out.println("Algum som genérico");
    }
}

// Subclasse (classe derivada)
class Cachorro extends Animal {
    void latir() {
        System.out.println("Latindo...");
    }
}

public class ExemploHeranca {
    public static void main(String[] args) {
        Cachorro meuCachorro = new Cachorro();
        meuCachorro.fazerSom(); // Método da superclasse
        meuCachorro.latir();    // Método da subclasse
    }
```

}
```

**Herança em C++:**

Em C++, a herança é também realizada através da palavra-chave `class`, mas a relação é definida com `:`. Além disso, diferentes tipos de herança (pública, privada, protegida) podem ser especificados.

**Exemplo em C++:**

```cpp
// Superclasse (classe base)
class Animal {
public:
    void fazerSom() {
        std::cout << "Algum som genérico" << std::endl;
    }
};

// Subclasse (classe derivada)
class Cachorro : public Animal {
public:
    void latir() {
        std::cout << "Latindo..." << std::endl;
    }
};

int main() {
    Cachorro meuCachorro;
    meuCachorro.fazerSom(); // Método da superclasse
    meuCachorro.latir();    // Método da subclasse

    return 0;
}
```

**Outros Exemplos:**

A seguir, exemplos adicionais demonstrando herança em Java e C++:

1. **Herança Simples:**
   ```java
   class Veiculo {
       int velocidade;
       void acelerar() {
           System.out.println("Acelerando...");
       }

```
}

class Carro extends Veiculo {
    void abrirPorta() {
        System.out.println("Abrindo porta do carro...");
    }
}
```

2. **Herança Múltipla em C++:**
```cpp
class A {
public:
    void metodoA() {
        std::cout << "Método A" << std::endl;
    }
};

class B {
public:
    void metodoB() {
        std::cout << "Método B" << std::endl;
    }
};

class C : public A, public B {
public:
    void metodoC() {
        std::cout << "Método C" << std::endl;
    }
};
```

3. **Herança com Construtores:**
```java
class Pai {
    Pai() {
        System.out.println("Construtor da classe Pai");
    }
}

class Filho extends Pai {
    Filho() {
        System.out.println("Construtor da classe Filho");
    }
}
```

4. **Herança com Sobrescrita de Método em Java:**
   ```java
   class Forma {
       void desenhar() {
           System.out.println("Desenhando uma forma");
       }
   }

   class Circulo extends Forma {
       @Override
       void desenhar() {
           System.out.println("Desenhando um círculo");
       }
   }
   ```

5. **Herança com Sobrecarga de Método em C++:**
   ```cpp
   class Base {
   public:
       void mostrarMensagem() {
           std::cout << "Mensagem da classe Base" << std::endl;
       }

       void mostrarMensagem(int numero) {
           std::cout << "Número: " << numero << std::endl;
       }
   };
   ```

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

Ao declararmos uma variável em Java, não estamos exatamente criando um ponteiro, mas sim uma referência. Em Java, todas as variáveis de objeto são referências, e a alocação de memória ocorre dinamicamente no heap quando usamos a palavra-chave 'new'. Essa palavra-chave reserva espaço para o objeto e retorna uma referência à área de memória alocada.

Diferenças importantes entre referências em Java e ponteiros em C++:

- **Operações Aritméticas de Ponteiros:**

  Em C++, ponteiros suportam operações aritméticas; em Java, referências não permitem isso.

- **Manipulação Direta de Memória:**

Em C++, é possível manipular diretamente a memória; em Java, a JVM gerencia automaticamente, impedindo acesso direto.

- **Gestão de Memória:**

  Java usa coletor de lixo para gerenciar memória automaticamente; em C++, é responsabilidade do programador.