Alunos: Giovani Zanella da Maia, Sofia Effting e Eduardo Wagner.

1) Descreva seu entendimento acerca dos conceitos de Classe, Objeto é Instância.
Classe:

- Uma classe é uma definição de um tipo de objeto. Ela descreve a estrutura e o comportamento dos objetos que serão criados a partir dela.
- Classe é um tipo definido pelo usuário. Classe consiste em um grupo de dados e funções membros.
- Uma classe é uma especificação, como um modelo ou plano para criar objetos que compartilham características comuns.

Objeto:

- Um objeto é algo do mundo real que eventualmente será descrito por uma classe.
- Em programação orientada a objetos, um objeto é uma entidade concreta que será representada na linguagem por uma classe.
- Um objeto possui atributos (estados e propriedades) e comportamentos (ações, métodos).

Instância:
Em programação orientada a objetos, uma instância é, em essência, uma ocorrência de objeto. A ideia é que quando você cria um objeto a partir de uma classe, você está criando uma "instância" específica dessa classe. Portanto, uma instância é uma ocorrência de um objeto que segue a estrutura e o comportamento definidos pela classe.

Em resumo:

- Uma instância de um objeto é, geralmente, acessada (eventualmente) através de uma variável que possui um tipo da classe que representa o objeto.
- Um objeto é uma instância de uma classe. Ele é criado com base na definição da classe e representa uma entidade específica com seus próprios atributos e comportamentos.
- Uma "classe" é um modelo ou um plano que define a estrutura (atributos) e comportamento (funções membros / métodos) que os objetos terão.
- Uma "instância" é simplesmente outra forma de se referir a um objeto criado a partir de uma classe.
- Cada instância criada a partir de uma classe é uma entidade independente com seu próprio estado e comportamento.

2) Qual a importância da especificação de métodos construtores e destrutores (ou destruidores) na programação orientada a objetos.
Os construtores são como "preparadores" que arrumam tudo para os objetos começarem a funcionar, organizando espaço na memória e ajustando configurações iniciais. Já os destrutores entram em ação quando um objeto não é mais necessário, garantindo que recursos como memória sejam devolvidos corretamente.

3) A despeito das similaridades quanto ao uso, qual a principal diferença presente nos conceitos de classe abstrata e interface.

Classe Abstrata:

Propósito: Destinada a ser a base para outras classes. Ela pode conter implementações concretas de alguns métodos e declarar outros como abstratos, que devem ser implementados pelas subclasses.
Estado e Comportamento: Pode conter estados (atributos) e comportamentos (métodos) concretos e abstratos.
Herança: As subclasses que herdam de uma classe abstrata devem implementar métodos abstratos e podem sobrescrever os concretos. Uma classe pode herdar de apenas uma classe abstrata (herança única).
Utilização: Mais adequada quando há uma relação "é um tipo de" e quando as classes derivadas compartilham uma estrutura ou comportamento comum.
Interface:

Propósito: Atua como um contrato para as classes que a implementam, estabelecendo um conjunto de métodos que essas classes devem implementar.
Estado e Comportamento: Tradicionalmente, não pode conter estado (atributos concretos) e apenas declara métodos abstratos. No entanto, em algumas linguagens como Java, interfaces podem conter métodos padrão (default) e métodos estáticos.
Herança: Uma classe pode implementar várias interfaces, proporcionando herança múltipla de tipo.
Utilização: Ideal quando diversas classes, que podem não ter uma relação hierárquica direta, precisam expor um comportamento comum.

4) Em que consiste o polimorfismo presente no conceito de programação orientada a objetos.

Polimorfismo se refere à capacidade de objetos de diferentes classes responderem de maneira única a chamadas de métodos com o mesmo nome. Isso significa que, em um sistema polimórfico, objetos de diferentes classes podem compartilhar uma interface comum e responder a essa interface de maneira específica à sua própria classe. O polimorfismo permite tratar objetos de classes diferentes de maneira uniforme, simplificando o código e tornando-o mais flexível.

Para entender o polimorfismo, é importante considerar a herança e os métodos em POO:

1. **Herança:** Classes diferentes podem herdar características (atributos e métodos) de outras classes. Uma classe filha herda características da classe mãe.
2. **Métodos:** As classes podem ter métodos com o mesmo nome, mas cada classe pode implementar esses métodos de maneira diferente.

Agora, o polimorfismo entra em cena:

- O polimorfismo permite que você use um único nome de método ou função para realizar operações em diferentes objetos de classes relacionadas.
- O comportamento específico do método é determinado pelo tipo do objeto que está sendo manipulado, e não pelo tipo da variável que o referencia.
- Em outras palavras, você pode chamar o mesmo método em diferentes objetos e obter resultados diferentes com base na classe real desses objetos.

- **Polimorfismo de Sobrescrita (Override):**
  Este tipo de polimorfismo ocorre quando uma subclasse fornece uma implementação específica de um método que já foi definido em sua classe base (superclasse). A ideia é substituir a implementação original pelo comportamento personalizado da subclasse. Isso permite que você chame o mesmo método em diferentes objetos, mas obtenha comportamentos diferentes, dependendo da classe do objeto.
- **Polimorfismo de Sobrecarga (Overloading):**
  Este tipo de polimorfismo ocorre quando uma classe tem múltiplos métodos com o mesmo nome, mas com diferentes tipos de parâmetros. Isso permite que você chame o mesmo método com diferentes argumentos, e o sistema de programação orientada a objetos determinará qual versão do método deve ser executada com base nos argumentos passados.
- **Polimorfismo de Interface:**
  Este tipo de polimorfismo ocorre quando várias classes implementam a mesma interface ou herdam da mesma classe abstrata, mas cada uma fornece sua própria implementação para os métodos definidos na interface ou classe abstrata. Isso permite tratar objetos de diferentes classes de maneira uniforme, contanto que sigam a mesma interface.

5) No paradigma orientado a objetos, o conceito que assegura que não haverá acesso direto aos dados é o encapsulamento.

6) Em se tratando da linguagem C++, no que consiste um ponteiro e qual a finalidade do tipo void.

Em C++, um ponteiro é uma variável que armazena o endereço de memória de outra variável. Em outras palavras, ao invés de armazenar um valor diretamente, um ponteiro armazena a localização na memória onde o valor pode ser encontrado. Isso oferece a capacidade de manipular indiretamente dados, o que é útil para operações avançadas e alocação dinâmica de memória.

O tipo void é usado para indicar a ausência de tipo. Ele pode ser utilizado em diferentes contextos para expressar que algo não possui um tipo específico, como por exemplo o retorno de um método ou como um ponteiro genérico.

7) Um importante conceito dentro do paradigma orientado a objetos é o de herança. Em qual circunstância pode ocorrer a herança múltipla.

A herança múltipla na programação orientada a objetos ocorre em circunstâncias onde uma classe precisa combinar características e comportamentos de mais de uma classe base. Isso é útil quando uma nova classe precisa herdar e integrar funcionalidades distintas de diferentes linhagens

8) Na orientação a objetos, um recurso que permite inicializar os atributos e é executado de forma automática quando um novo objeto é criado é conhecido como construtor.

9) No paradigma orientado a objetos, os relacionamentos entre classe base (superclasse) e classe derivada é realizado a partir de herança.

10) **O** Polimorfismo de Sobrescrita (Override) **de um método é o recurso por meio do qual uma classe derivada reescreve o método da classe-base, a fim de atender a alguma particularidade.**

11) **O que são os streams e quais os tipos disponíveis na linguagem C++.**

Stream é uma abstração que deixa mais fácil a entrada e saída de dados. E são implementados como classes na biblioteca padrão, ex: STL(Standard Template Library) e deixa de forma consistente para interagir com diferentes dispositivos de entrada e saída, como strings, console, etc. Os tipos são istream(entrada), ostream(saída) e iostream(entrada/saída stream).

12) Cite duas diferenças entre a classe string e strings designadas por variáveis do tipo char.

- Tamanho Dinâmico: A classe string em muitas linguagens de programação permite que você manipule strings de tamanho variável dinamicamente. Em contraste, as strings designadas por variáveis do tipo char geralmente têm um tamanho fixo e, se você precisar de uma string maior, será necessário alocar mais espaço e lidar com realocações manualmente.
- Funcionalidades e Métodos: A classe string frequentemente oferece um conjunto de métodos e funcionalidades embutidos que facilitam a manipulação de strings. Ao usar variáveis do tipo char para strings, você muitas vezes precisa recorrer a funções da biblioteca padrão para realizar operações básicas.

13) Programação orientada a objetos é um paradigma de programação baseado no conceito de objetos. Considerando as estruturas utilizadas na programação orientada a objetos, relacione adequadamente os conceitos a seguir, com as colunas abaixo.
1. Classe; 2. Objeto; 3. Método; 4. Atributo.

(**3**) Define-se dentro de uma classe para descrever o comportamento de um objeto. Programadores podem reutilizar ou manter a funcionalidade encapsulada dentro de um objeto;
(**1**) Tipo de dados definido pelo usuário que atua como um modelo para objetos, atributos e métodos individuais;
(**4**) Define-se na classe e representa o estado de um objeto; pertence à própria classe
(**2**) Instância de uma classe criada com dados definidos; pode corresponder a objetos do mundo real ou a uma entidade abstrata;